VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Bakalářská práce

2012 Daniel Žažo

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Synchronizace personálních dat přes internet

Internet Personal Data Synchronization

2012 Daniel Žažo

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Daniel Žažo**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Synchronizace personálních dat přes internet
Internet Personal Data Synchronization

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit aplikaci pro synchronizaci osobních dat, především souborů z kancelářských aplikací, přes Internet. Aplikace bude monitorovat určený adresář a její vzdálená část musí být spustitelná na bežně dostupných hostingových službách. Dalé musí umožňovat ochranu synchronizovaných dat před jejich zneužitím. Práce musí obsahovat především:

1. Analýzu problému synchronaizace dat z více zdrojů, tj. z více zařízení připojených k internetu. Tato synchrnoizace nemusí probíhat současně.
2. Analýzu problém distribuce dat v síťovém prostředí s firewallem (NATem).
3. Výběr vhodné metody pro synchronizaci přes internet.
4. Implementaci aplikace v některém běžném (i skriptovacím) jazyce.
5. Testování aplikace a prezentaci výsledků.

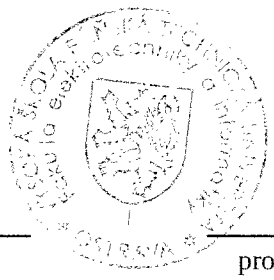
Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marian Mindek, Ph.D.**

Datum zadání: 19.11.2010
Datum odevzdání: 04.05.2012



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna: Daniel Žažo .....

Daniel Žažo

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Marianu Mindekovi, Ph.D. za cenné rady spojené s tvorbou této práce, které mi moc pomohly. Dále bych chtěl poděkovat rodině a všem, kteří mě podporovali. Děkuji.

Abstrakt

Každý uživatel počítače má své návyky a potřebuje pro svou potřebu určitý software, dokumenty apod. S růstem oblasti informačních technologií dochází k tomu, že uživatelé počítačů mohou pracovat na více počítačích a může nastat požadavek, že chtějí mít určitá data na všech počítačích stejná. Je tedy možné data z jednoho počítače nahrát na přenosné médium, a pak podle své potřeby data z média nahrát do jiného počítače, a takto jsou data určitým způsobem synchronizována, ovšem tento způsob synchronizace dat může být časem nepohodlný. Pokud jsou ale počítače, které chce uživatel synchronizovat, připojené k internetu, tak existují dnes programy, které tento problém řeší pomocí synchronizace přes internet. A právě synchronizaci dat přes internet, které je věnována tato práce, je cílem ukázat, jaké jsou možnosti data takto synchronizovat, s jakými problémy synchronizace přichází do styku a vlastní implementace aplikace pro synchronizaci dat přes internet.

Klíčová slova: Synchronizace, klient, server, internet, Java, Swing, Vaadin, MySQL, XML

Abstract

Each computer user has his own habits and uses certain software, documents, etc. for his own needs. It happens thanks to the increase in information technology that computer users can work on more computers and it can happen that they want to have the same data on all computers. It is possible to record the data from one computer to a removable medium and then according to needs to upload the data from the medium to another computer so that the data is synchronized in a certain way, but of course this method of synchronization will become uncomfortable after some tire. If theme computers, which the user wants to synchronize, connected to the internet, so nowadays there are programs that solve this problem by synchronizing them over the internet. This work is devoted to the data synchronization over the internet. Its main aim is to show these things: The data synchronization possibilities, the problems of synchronization and own application implementations of synchronization over the internet.

Keywords: Synchronization, client, server, internet, Java, Swing, Vaadin, MySQL, XML

Seznam použitých symbolů a zkratek

AES	- Advanced Encryption Standard
API	- Application Programming Interface
BIGINT	- Big Integer
BLOB	- Binary Large Object
DAO	- Data Access Object
DB2	- dBase 2
DES	- Data Encryption Standard
ER	- Entity-Relationship
GB	- Giga Byte
GFS	- Global File System
GWT	- Google Web Toolkit
IP	- Internet Protocol
JDBC	- Java Database Connectivity
JVM	- Java Virtual Machine
LONGBLOB	- Long Binary Large Object
MB	- Mega Byte
MBR	- Master Boot Record
MD5	- Message-Digest 5
MEDIUMBLOB	- Medium Binary Large Object
MS SQL	- Microsoft Structured Query Language
MySQL	- My Structured Query Language
NAT	- Network Address Translation
NFS	- Network File System
OS	- Operating System
TFS	- Team Foundation Server
TINYBLOB	- Tiny Binary Large Object
TripleDES	- Triple Data Encryption Standard
VARCHAR	- Variable Character Field
WAR	- Web Application ARchive
XML	- eXtensible Markup Language
iOS	- iPhone Operating System

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisu zdrojového kódu

1 Úvod

S narůstajícím časem přichází do styku s počítačem mnohem více jeho uživatelů a ne každý uživatel pracuje pouze s jedním počítačem, ať už je to z různých důvodů. Někdy může nastat nebo dokonce je nutností potřeba mít data na více počítačích stejná a to znamená, že uživatel je donucen data nějakým způsobem sjednotit (synchronizovat).

Synchronizování dat je dnes důležitou součástí v oblasti informačních technologií a řadě z nás to ani nedochází. Který projekt v této oblasti s větším počtem vývojářů by byl schopen fungovat bez synchronizování dat? Ostatně nemusí to být projekt pouze s větším počtem vývojářů, ale stačí tým složený pouze ze dvou osob a ti musí mezi sebou spolupracovat, pokud to je potřeba. Synchronizovat data mezi sebou pomocí technologie Bluetooth, kabelového spojení nebo nějakého přenosného média je nemyslitelné. Navíc může nastat situace, že oba budou pracovat na stejné části projektu, a pak při následné synchronizaci může nastat konflikt a která verze, od kterého vývojáře je správná? Najednou nastal nechtěný problém mezi dvěma programátory a stačí vzít v úvahu tým složený minimálně z deseti programátorů, což je zcela běžné a takové synchronizování dat s předáváním média, použití kabelů, či použití Bluetooth nepřipadá v úvahu. Proto byly vytvořeny nástroje, které řeší práci na projektech v týmu a zajišťují synchronizaci dat i tam.

Dnes už nejsou data synchronizovaná pouze v této oblasti, ale s vývojem technologií jsou vyvíjeny i jiné oblasti jako jsou například mobilní zařízení. Některé mobilní telefony jsou v dnešní době už tak výkonné, že poskytují velmi rozsáhlé možností. Mnohdy je dost užitečné synchronizovat data mezi těmito zařízeními a počítači. Přitom data nemusí být vždy data v počítači v podobě souborů, ale můžou to být kontakty, kalendáře, e-maily a různé soubory všeho druhu. A tak díky těmto aplikacím můžeme mít stejná data na počítači, notebooku, tabletu nebo mobilním telefonu, či jiném podporovaném zařízeni.

Nejčastěji je synchronizace dat spojena v souvislosti synchronizace dat mezi více počítači, a právě této problematice je věnována tato bakalářská práce, jež je rozdělena do tří majoritních kapitol. Kapitola druhá je věnována aplikacím, které jsou aktuálně k dispozici na trhu a synchronizují osobní data uživatele. V této kapitole jsou analyzovány a popsány nejznámější programy, pro synchronizaci dat přes internet. Ve třetí kapitole jsou rozebrány problémy, které je potřeba brát v úvahu při synchronizaci a čtvrtá kapitola je věnována vlastní implementaci takové aplikace.

2 Přehled aplikací na trhu

V současnosti je na trhu řada aplikací, které synchronizují různá data odlišnými způsoby. Jak už to bývá, každá aplikace má své pro a proti a uživatel musí vyzkoušet různé aplikace, než narazí na tu, která mu vyhovuje, protože každý má svůj styl práce a používá věci tak, jak jsou mu blízké pro usnadňující práci. Tato práce je věnována osobní synchronizaci dat přes internet a touto synchronizaci je míněno synchronizování veškerých kancelářských i jiných dokumentů, hudebních souborů, fotografií v odlišných formátech, konfiguračních souborů, zkrátka vše, co lze považovat v počítači za soubor. K tomuto účelu slouží programy, jež data nahrávají automaticky na server a vytváří tak kopii souborů, které máme v počítači nebo mobilním zařízeni.

V této kapitole jsem analyzoval a popsal, jaké má uživatel možností, které mu nabízí nejznámější a nejpoužívanější aplikace, které jsou aktuálně na trhu a synchronizují osobní data uživatelů. Tato analýza mi zároveň sloužila jako inspirace pro vývoj vlastního řešení. Pro otestování aplikací jsem používal operační systém *Windows 7*, takže například velikosti instalačních balíků, které uvádím, mohou mít pro různé platformy odlišné velikosti, pokud je aplikace podporují.

2.1 Dropbox

Velmi populární aplikace *Dropbox* [1] je neustálé vyvíjen a patří mezi nejlepší. Přímo na stránkách aplikace je možné shlédnout video tutoriál, kde je v pár bodech ukázáno, jak zřídit účet a používat program. Uživatel má možnost vytvořit účet přímo na stránkách aplikace nebo ho může vytvořit až při instalaci programu. Uživatelským jménem musí být e-mailová adresa uživatele. Při instalaci je možnost opět shlédnout návod v pár bodech a ke konci instalace je vytvořena složka *Dropbox*, jejíž obsah je synchronizován. Pokud někomu nebude vyhovovat název složky, může být uživatelem změněn podle jeho uvážení, ale musí uzavřít všechny soubory, které jsou v této složce umístěny a potom program úplně vypnout.

Po vložení, vytvoření či editování nějakého souboru v této složce začne aplikace v defaultním okamžitě nahrávat data na server a vytváří tak kopií synchronizované složky, z čehož vyplývá, že je nutností být připojený k internetu. Aplikace nemusí synchronizovat pouze jednu složku, ale existuje možnost přiřadit i jiné složky pro synchronizování, což je provedeno automaticky tak, že nová složka je přesunuta do složky *Dropbox* a namísto původní složky je vložen link na její aktuální umístění v nové složce. Tato akce je prováděna *Dropboxem* automaticky, uživatel pouze vybere, jakou další složku chce synchronizovat.

Síla *Dropboxu* není pouze v synchronizování složek a souborů, ale vyniká i tím, že data uchovávaná na serveru, kde je vytvořena kopie této složky, slouží jako záloha a co víc, tak k těmto datům je možno přistupovat přes webové rozhraní, pomocí něhož je možné data stáhnout, kdekoli bude třeba, stačí jen projít přihlašovacím procesem. *Dropbox* přes toto všechno ukládá ještě historii veškerých události a nastane-li, že uživatel omylem smazal data ze složky, může je kdykoli obnovit pomocí webového rozhraní, ale to jen do určité doby, po kterou jsou data archivována.

Další příjemnou funkcí je sdílení určité složky s jiným uživatelem, který také používá *Dropbox*. Funguje to tak, že uživatel vybere složku ve svém *Dropboxu*, kterou chce sdílet s jiným uživatelem a programem je vyzván o vyplnění mailu onoho uživatele a o vyplnění textu, který uživateli přijde. Uživatel bude dotázán, zda chce složku sdílet. Pokud uživatel souhlasí, má každý z nich ve své složce

Dropbox složku, kterou mají společnou a navzájem synchronizovanou. Tato možnost není vázaná jen na dva uživatele, protože počet uživatelů, kteří můžou sdílet určenou složku, není omezen.

Aplikace *Dropbox* už není dnes pouze na počítače, nýbrž je možné data sdílet i mezi mobilními zařízeními s odlišnými operačními systémy, jakou jsou *Android*, *iOS* nebo *BlackBerry OS*. Velikost místa, které je poskytováno je 2 GB v neplacené verzi, ovšem je nabízena možnost místo navýšit až na 16 GB za pozvání nového uživatele *Dropboxu*. Za každého uživatele, který je zaregistrován pomocí odkazu, jež mu byl zaslán původním uživatelem, tak obdrží 500 MB navíc. Z toho plyne, že je možné pozvat až 28 osob, za které je obdržen bonus. Pokud by některý uživatel chtěl využívat více místa, než je 2 – 16 GB, tak potom existují i placené verze.

Základní vlastnosti:

- multiplatformnost (*Android*, *BlackBerry*, *Linux*, *Mac*, *Windows*)
- webové rozhraní
- jazyk: angličtina
- velikost instalačního balíčku pro Windows – 14,3 MB
- kapacita úložiště: 2 – 16 GB – free; 50 GB - 9,99$ / měsíčně; 100 GB – 19,99$ / měsíčně

2.2 SugarSync

Alternativou k *Dropboxu* může být program *SugarSync* [2], který je mu dosti podobný. *SugarSync* ale má pochopitelně své specifické vlastnosti. Na svém webu je na první pohled možné zpozorovat velké kapacity úložiště, které je zdarma a to je 5 GB, což je víc jak dvojnásobek oproti *Dropboxu* Na webu je možnost shlédnout video tutoriál, který uživatele seznámí s použitím a konfigurací programu.

Při instalaci je uživatel vyzván k zadání údajů, popřípadě k vytvoření nového účtu, pokud ho ještě nevlastní. Dále je pak možnost vybrat ikonu složky, která bude synchronizována. Ikony jsou v pestrých barvách, jednoduché a pohledné. Dalším krokem je výběr složek pro synchronizaci. Defaultně jsou v programu vybrány složky Plocha, Dokumenty, Obrázky, Hudba a Videa. Tyto složky je možno odškrtnout nebo popřípadě přidat další. Následuje popis programu a seznam mobilních zařízení, které jsou podporována. Aplikace *SugarSync* pro mobilní zařízení je pro platformy *Android*, *iPhone*/*iPad*, *BlackBerry*, *Symbian* a *Windows Mobile*. Nakonec je programem vytvořena složka *Magic Briefcase*, která slouží pro synchronizaci.

Webové rozhraní je *Dropboxu* velmi podobné, odlišné skoro jen v barevném provedení a logu značky. U *SugarSync* je ale web je o něco obsáhlejší, než je tomu u *Dropboxu* a nabízí řadu možností navíc. Ve svých vlastnostech pro nalákání nových uživatelů nabízí *SugarSync* online podporu. Ovšem živý chat funguje pouze v uvedených hodinách, a tak někdy při potížích nezbývá, než počkat, požádat o pomoc komunitu nebo odeslat formulář a počkat na odpověď. Zálohování dat i po smazání souboru či složky *SugarSync* není cizí, byť to platí taky pouze po určitou dobu.

Nabízeny jsou zde i odměny za přilákání nových uživatelů a to navýšení kapacity o 500 MB za každý nový účet. Toto navyšovaní je zastaveno až na 32 GB, pokud máme zdarma účet. Pokud uživatel vlastní placenou verzi, tak je velikost bonusů neomezena. V tomto ohledu *Dropbox* celkem ztrácí, vezmeme-li v potaz tento parametr. V celkovém srovnání s *Dropboxem* je *SugarSync* uživatelsky přívětivější a kvóta, kterou nabízí je mnohem výhodnější, ať už je srovnáván účet zdarma nebo placená verze. Vytknout mu je možno to, že postrádá podporu pro operační systém *Linux*, čímž je komunita uživatelů používající *Linux* nucena zvolit *Dropbox* nebo jinou alternativu.

Základní vlastnosti:

- multiplatformnost (*Android*, *BlackBerry*, *Mac*, *Symbian*, *Windows*)
- webové rozhraní
- jazyk: angličtina
- velikost instalačního balíčku pro Windows – 16,4 MB
- kapacita úložiště: 5 – 32 GB – zdarma; 30 GB - 4,99$ / měsíčně; 60 GB – 9,99$ / měsíčně, 100 GB – 14,99$ / měsíčně

2.3 Synplicity

Pokud někdo nebude spokojeny s výše uvedenými aplikacemi, tak může zvolit jinou a variantu a tou je například aplikace *Synplicity* [3], která disponuje zajímavou funkcí a to je propojení s dokumenty *Google Docs*. Ovšem používat synchronizaci ve verzi zdarma až pro dva počítače a mobilní zařízení o dostupné kapacitě 2 GB je dost velké omezení a nedosahuje takových kvalit jako již uvedené aplikace. Z webových stránek programu *Synplicity* můžete aplikaci stáhnout až po zaregistrování a následném přihlášení na účet.

Z tohoto faktu odpadá možnost vytvoření účtu při instalaci klientské aplikace, ale uživatel je rovnou vyzván, aby zadal uživatelské údaje pro ověření. Po tomto kroku je potřeba zvolit ze dvou možností, jaké data je potřeba synchronizovat. První možností je volba synchronizace dat taková, že na všech synchronizovaných zařízeních bude poslední verze a poslední verze bude uchovávána online na serveru. Druhou možností je plná kontrola nad synchronizovanými daty. Byla tedy zvolena první možnost, aby chování bylo nastaveno stejně, jako tomu bylo v již otestovaných aplikacích a dále po pár potvrzeních je aplikace velmi rychle nainstalovaná.

Synchronizovaná složka byla automaticky zvolena při instalaci a uživatel jí nemůže změnit. Automaticky navolené složky jsou Dokumenty uživatele a Plocha jeho počítače. Tyto složky může uživatel ze synchronizace odškrtnout a popřípadě přidat další. Otestováním bylo zjištěno, že *Synplicity* nesynchronizuje vše. Kupříkladu zástupci programů na ploše jsou přeskakovány a považovány za zbytečné při synchronizaci, oproti výše uvedeným aplikacím, které synchronizují i tyto typy souborů, tudíž je potřeba dávát pozor na různé typy souboru u tohoto programu.

Nastavení Synplicity je jednoduché, ale vzhledově zcela odlišné od předešlých. Synplicity disponuje taktéž webovým rozhraním, které je zcela odlišné od předchozích, ale uživatelský velmi přívětivé a jednoduché. Za pozvání nových uživatelů je bonus 1 GB kapacity navíc, ale tento bonus bude přidán nanejvýš 3x, díky čemuž je možné získat až 5 GB, kde navyšovaní končí, a pak nezbude nic jiného než pořídit placenou verzi. Za zmínku pak stoji podpora tabletu *Kindle Fire* (upravený *Android*).

Základní vlastnosti:

- multiplatformnost (*Android*, Mac, *Windows*, *Kindle Fire*)
- webové rozhraní
- jazyk: angličtina
- velikost instalačního balíčku pro Windows – 2,1 MB
- kapacita úložiště: 2 – 5 GB – zdarma; 50 GB - 15$ / měsíčně; bez limitu – 45$ / měsíčně

2.4 Windows Live Mesh

Velkým konkurentem všem uvedeným v oblasti synchronizace osobních dat je *Windows Live Mesh* [4], který je aktuálně součástí balíčku *Windows Live Essentials 2011*. Z názvu je patrné, že produkt je od firmy *Microsoft*, a tak je možné předem očekávat podporu pouze v zařízeních s operačním systémem od této firmy, ovšem *Windows Live Mesh* podporuje i počítače s operačním systémem *Mac*, ale pouze v novější verze tohoto systému.

Po bezproblémovém stáhnutí programu *Live Mesh* je nutno dávat pozor při instalaci. V prvním kroku je možnost zvolit instalaci všech součástí balíčku *Live Essentials 2011* nebo vlastní výběr programů tohoto balíčku, které uživatel vybere. Jelikož součástí zmiňované sady je například poštovní klient *Microsoft Outlook Express* nebo *Messanger*, tak by počítač mohl být zbytečně zpomalován a zatěžován těmito programy nebo přinejmenším by bylo zbytečně zabráno místo na disku, pokud je uživatel nepotřebuje. Je tedy zvolena vlastní instalace a na dalším okně je nutno odškrtnout všechny produkty a je ponechán pouze požadovaný Live Mesh. Poté instalační program stahuje neznámý počet dat ke své instalaci a po jejím úspěšném dokončení je možné program spustit.

Po spuštění programu je možno vidět pár základních možností důležité pro synchronizaci. Je možné vybrat více složek pro synchronizaci a při výběru každé složky je potřeba nastavit, které zařízení budou složku synchronizovat. Také je uživateli nabízeno synchronizovat záložky z Internetu Exploreru nebo dokumenty uživatele, tyto možností stačí jen zvolit.

Live Mesh disponuje webovým rozhraním, které je vcelku jednoduché, snadno ovladatelné a díky nabízeným 5 GB volného prostoru, jež disponuje, tak je konkurentem již zmíněných programů. Oproti těmto programům *Live Mesh* není po instalaci integrován do průzkumníku a klikneme-li pravým tlačítkem myši na soubor nebo složku, tak nemáme možnost zvolit synchronizaci tohoto souboru či složky, jako tomu je u předešlých konkurentů.

Základní vlastnosti:

- multiplatformnost (*Mac*, *Windows*)
- webové rozhraní
- jazyk: angličtina, čeština a mnoho dalších
- velikost instalačního balíčku pro Windows – 1,2 MB + neznámý počet stáhnutých dat při instalaci
- kapacita úložiště: 5 GB – zdarma

2.5 Shrnutí

Uvedené synchronizační programy jsou natěšeny s přibývajícím časem narůstající oblibě a uživatelé mohou vybírat na základě svých požadavků. Pokud někdo používá pouze operační systém *Windows* popřípadě *Mac*, tak pro své podmínky a nabízené možnosti je jasnou jedničkou *SugarSync*, a to především díky nabízených až 32 GB volného úložiště zdarma. Ovšem používá-li uživatel *Linux*, musí sáhnout po *Dropboxu* nebo hledat jiné alternativy. Vybrání synchronizačního programu tedy závisí, jaké zařízení s jakými operačními systémy uživatel používá. Taky je důležitým faktorem, jak uživatel s jednotlivými programy pracuje. Pro porovnání základních vlastností byly vytvořeny 2 tabulky (tabulka 1 a tabulka 2).

	Zdarma	Max. bonus	Čeština	Velikost instalačního balíku
Dropbox	2 GB	14 GB	Ne	14,3 MB
SugarSync	5 GB	27 GB	Ne	16,4 MB
Synplicity	2 GB	3 GB	Ne	2,1 MB
Live Mesh	5 GB	0 GB	Ano	1,2 MB + neznámý počet

Tabulka 1: Porovnání vlastností synchronizačních programů

	Windows	*Linux*	*Mac*	*Android*	*BlackBerry*	*Symbian*
Dropbox	Ano	Ano	Ano	Ano	Ano	Ne
SugarSync	Ano	Ne	Ano	Ano	Ano	Ano
Synplicity	Ano	Ne	Ano	Ano	Ne	Ne
Live Mesh	Ano	Ne	Ano	Ne	Ne	Ne

Tabulka 2: Tabulka ukazující podporu různých operačních systémů

3 Analýza problémů synchronizace dat přes internet

Synchronizace dat je proces, který má za úkol udržovat stejná data na všech úložištích, která jsou vybrána pro synchronizaci. Je-li mluveno o synchronizaci založené na bázi souborů, tak tu lze dále rozdělit do 4 kategorií [5]:

- **Synchronizace souborů** - obvykle používaná pro osobní synchronizaci mezi více zařízeními (*Dropbox*, *SugarSync*).
- **Kontrola verzí** - užíváno především pro synchronizací dat mezi více osobami (*TFS*, *Subversion*).
- **Distribuované souborové systémy -** zajišťují více verzí synchronizovaných souborů, podobně jako kontrola verzí, ale nedisponují tolika možnostmi.
- **Zrcadlení -** zrcadlo je přesnou kopií.

Při synchronizování dat přes internet je spojena řada problémů [6], ač to na první pohled není patrné. Sjednotit data byť pouze na 2 počítačích obnáší řadu problémů a za dosažením stejné kopie na obou zařízeních je skryta dlouhá cesta. Dále jsem popsal možné problémy, s kterými jsem přišel do styku nebo které připadají v úvahu při synchronizaci dat.

3.1 Setříděnost dat

Problém setříděných dat souvisí s tím, jakým způsobem jsou data synchronizována. Data je možno obecně ukládat dvěma různými způsoby a to setříděně nebo nesetříděně. Z těchto uložení dat pak vychází jednotlivé aplikace, které používají svůj algoritmus pro synchronizaci. Synchronizačních algoritmů je celá řada a vždy je potřeba správně zvolit ten, který je pro danou situaci nejvhodnější.

3.1.1 Nesetříděné data

Nesetříděná nebo také neuspořádána data jsou známá také pod názvem *problém nastavení smíru* za účelem stejných souborů v synchronizovaných zařízeních. Problém je vyjadřován jako pokus o výpočet symetrického rozdílu vztahem $S_A \otimes S_B = (S_A - S_B) \cup (S_B - S_A)$ mezi soubory S_A a S_B bitových čísel. Mezi nejpoužívanější způsoby řešení nesetříděných dat je používáno:

- **Hromadný přenos** – přenesení všech dat a poté lokální porovnání.
- **Časové razítko** – všechny změny jsou označeny časovým razítkem a následně jsou synchronizovány pouze ty data, které mají časové razítko novější než původní.
- **Matematická synchronizace** – data jsou považována za matematické objekty a následná synchronizace koresponduje s matematickými procesy.

3.1.2 Setříděné data

Této vlastnosti využívá drtivá většina souborových sytému na bázi synchronizace. Zde je potřeba udržet dva vzdálené řetězce sjednocené. Obvykle jsou tyto řetězce různé o pevný počet editací (editace, vložení, odstranění). Synchronizace pak snižuje počet editací mezi řetězci až do ideální vzdálenosti od nuly.

3.2 Konflikty

Nevyhnutelným problémem při synchronizaci jsou konflikty, které je potřeba brát v úvahu. Konflikt nastane v situaci, kdy je manipulováno s právě jedním souborem z více stran. Konflikt může nastat zcela běžně, což popisuji v následujících odstavcích. Popsal jsem zde i různé možnosti, jak možné konflikty vyřešit.

Uvažujme následující situaci. Uživatel v práci na počítači změní soubor, který je úspěšně pomocí synchronizačního programu nahrán na server. Doma bude opět na tomto souboru muset pracovat, a tak zapne doma počítač, ale nepostřehne, že mu zrovna nejde internet. Soubor už dříve měl vytvořený ve složce, a tak mu nic nebrání pokračovat v práci. Nevšimne si, že jeho poslední změna, kterou provedl v práci, tak není promítnuta v souboru a soubor dále edituje, ovšem aniž by to věděl, tak pracuje s neaktuální verzí souboru, protože aktuální verze je na serveru. Najednou počítač naváže spojení s internetem a synchronizační program bude chtít synchronizovat soubor, na kterém pracoval a teď i pracuje. Ovšem zjistí, že uživatel pracoval na souboru, který měl být před jeho prací synchronizovaný, ale nebyl. Nastal tedy konflikt, který je zapotřebí brát v úvahu a při synchronizaci musí být s ním počítáno. Řešení konfliktu má obecně 3 základní možností, jak ho vyřešit. Možností nabídne uživateli v době, kdy konflikt nastal nebo jsou konflikty řešeny automaticky na základě nastavení:

- Použít soubor, který mu nabízí server, jako posledně upraveny a přijít tak o data, která byla změněna doma.
- Použít aktuální soubor, na kterém uživatel pracoval doma a přijít tak o data, která změnil v práci.
- Soubor nějak sloučit. Jak, to už záleží na implementaci jednotlivých aplikací.

Tyto kroky lze v aplikacích někdy nastavit automaticky, ať uživatel není neustálé zatěžován dotazy, ale toto bych nedoporučoval, neboť tak lze snadno ztratit přehled. Někdy to ovšem může přijít vhod. Také nemusí být brán v potaz pouze jeden soubor od jednoho uživatele. Konflikty často nastávají u vývojářů a stačí k tomu jen, aby někdo například naformátoval zdrojový kód. Druhý vývojář, co pak bude chtít odeslat výslednou verzi na jednotné místo pro všechny, tak mu bude zahlášen konflikt a nabídnuta řada možností. Musí tedy jednotlivé změny projít a uvážit možností. V tomto případě by mohl označit svůj soubor, za výsledný, neboť předtím byl pouze kód zformátován a nic k němu nepřibylo, ovšem toto je jen výjimečný případ. Je tedy užitečné zabránit tomu, aby pracovalo více osob na stejném souboru, neboť to může přinášet problémy a práce je neefektivní.

3.3 Bezpečnost dat

S používáním internetu pro synchronizaci dat je uživatel vystavován jistému riziku, neboť jeho komunikaci může někdo odposlouchávat a určitě je nežádoucí, aby osobní dokumenty byly veřejně dostupné a mohl k nim mít kdokoli přístup. V horším případě může nastat velký problém, kdy data budou velmi citlivé a dostane je do rukou někdo nežádoucí, jež data zneužije. Komunikaci je tehdy vhodným způsobem zabezpečit, ať je nezíská jen tak někdo.

Je potřeba taky brát v úvahu, kde budou data fyzicky uložena, kdo k nim bude mít přístup a zabránit tak jejich odcizení či poškození. V případě, že nastane nejhorší varianta a médium s daty bude odcizeno, tak je dobré, aby data byla zabezpečena i na médiu a útočník, tak měl ztíženou práci data odcizit a v lepším případě i znemožněnou.

3.3.1 Bezpečnost komunikace

Komunikace je v oblasti informačních technologií zabezpečována pomocí šifrování, které spadá do oboru kryptografie. Šifrování má svůj dávný původ a dnes už existuje řada možností [7], jak data šifrovat. V oblasti informatiky jsou šifry děleny do 3 skupin, kde každá má své výhody i nevýhody použití.

Z pohledu bezpečností je nejméně bezpečné *symetrické šifrování*, které pro zašifrování i dešifrování používá stejný klíč. Šifer, jež využívá symetrického šifrování, je dnes celá řada a záleží vždy na konkrétním použití, který způsob zvolit. Mezi nejznámější symetrické šifrování patří:

- ***DES*** [8] – délka klíče činí pouhých 56 bitů, což je dnes už mnohdy nedostačující.
- ***TripleDES*** [9] – z názvu lze odvodit, že délka klíče bude až 168 bitů (3 x *DES*), což eliminuje nedostatek původní metody *DES*.
- ***AES*** [10] – náhrada výše uvedených metod, která používá pevnou délku klíče a to ve třech různých velikostech: 128, 192 a 256 bitů.

Dalším druhem šifrování je *asymetrické šifrování*, u něhož jsou používané dva klíče. První klíč je nazván veřejný, kterým jsou data zašifrována. Druhý klíč je jiný, jak první a jsou jim data dešifrována. Tento klíč je nazván privátním klíčem. Zástupců asymetrického šifrování je několik, ale v používání vyniká jedno a tím je:

- ***RSA*** [11] - algoritmus používán pro podepisování a šifrování, který je díky své délce klíče považován za bezpečný. Obvykle je používaná délka klíče 517 – 4096 bitů.

Posledním způsobem, jak data zašifrovat je tzv. *hybridní šifrování*, které je kombinací dvou předchozích a spojuje výhodou obou v jednom. Zabezpečení dat pak závisí na obou použitých šifrách, které byly použity.

3.3.2 Bezpečnost uložených dat

Je třeba pamatovat na to, že pro dobré zabezpečení dat je brát v úvahu i místo, kde jsou data ukládaná, protože tento fakt je hodně opomíjen, i když má velkou důležitost. Pokud je využíváno služeb, které poskytují různá úložiště dat, tak těžko zjistíme, kdo všechno bude mít přístup k datům po fyzické stránce. Z tohoto hlediska je nejlepší vytvořit vlastní server a provozovat vše na něm. Pro jistotu je data vhodné na disku zašifrovat. Jak to udělat, to už závisí na konkrétním případu, kterému bude majitel nakloněn. Možností jsou následující:

- **Šifrování disku** – pro šifrování disku je používán software, který šifruje i *MBR* a disk je celý zašifrován.
- **Šifrováno souboru** – soubory je možné šifrovat jednotlivě nebo lze šifrovat i různé části disku, ovšem pokud synchronizační programy již data šifrují, tak na médiích by měly být taky uložena.

3.3.3 Shrnutí

Je potřeba mít na mysli, že zašifrováním dat není zabráněno útočníkům je získat, pouze je znepříjemněna cesta k jejich zneužití. Data je ale nutné chránit a je potřeba používat šifrování a různá zabezpečení.

3.4 Uložení dat

Důležitým faktorem pro provoz služby na synchronizaci přes internet je způsob uložení dat a s tím jsou spojené otázky, které je dobře vyhodnotit. Data je možno uchovávat dvěma způsoby a to s použitím databáze nebo nějakého síťového souborového systému. Při uložení dat je nutné počítat s poškozením média a data je potřeba i zálohovat.

3.4.1 Zálohování

Při poškození média pro ukládání dat nebo jeho ztrátě je potřeba ztracená data obnovit. K tomu je nutno data zálohovat a zamezit případnému problému. Jak zálohovat data už záleží na konkrétní potřebě a i zde existuje řada možností [12], jakou variantu zvolit.

- **Typ zálohy** – zálohovat je možno strukturovaně, nestrukturovaně, inkrementálně, použitím nějakého typu zrcadlení nebo kombinace různých typu zálohy.
- **Médium pro uložení dat –** je možné použít takřka jakéhokoliv média od přenosných disků přes různé magnetické pásky, pevné disky nebo zálohovacího místa v sítí.
- **Režim zálohy** – z pohledu režimu jsou 2 základní typy. První možností je zálohovat za chodu serveru nebo po odstavení serveru. Z toho jsou i odpovídající názvy režimu zálohy a to *Online* nebo *Offline.*
- **Interval zálohování** – podle potřeby je rozhodnuto, v jakém čase a jak často zálohu vytvářet.
- **Automatizovanost** – aby bylo předcházeno lidskému selhání, tak je možno využívat automatické zálohy, pokud je to daná aplikace nabízí.
- **Umístění zálohovacího média** – například z důvodu požáru je vhodné mít zálohovací médium v jiné budově a mít tak záložní kopii v bezpečí od vzniklého nebezpečí.

3.4.2 Způsob uložení

Ukládat data v sítí lze obecně dvěma způsoby. První možností je použití klasické databáze nebo distribuovat data pomocí síťového souborového systému.

3.4.2.1 Použití databáze

Při výběru databáze je třeba brát ohled na používané technologie a jednotlivé požadavkům aplikace. Databáze by měla podporovat uložení dat v datovém typu *BLOB* [14], což slouží pro uložení souborů v jakémkoliv formátu. Nejznámější databáze tento formát podporují, a tak je třeba jen zjistit, jak je datový typ v konkrétní databázi nazván a používán, protože *BLOB* může mít i jiný název jako například *BYTEA, VARBINARY* apod. Z těchto názvu je možno vyvodit, že soubory jsou ukládány jako binární data. S použitím databáze je pak potřeba zvolit vhodné práva pro správu dat, aby nedocházelo k nežádoucím problémům. Mezi nejznámější databáze patří *Oracle*, *MySQL*, *MS SQL Server*, *PostgreSQL*, *DB2* nebo *Derby*.

3.4.2.2 Použití síťového souborového systému

Podobně jako je to u použití databáze, tak i zde je potřeba brát ohled na používané technologie, které umožňuji používat souborovým systémem v sítí. Souborový systém [14] v sítí je normální počítač, kde soubory a složky leží na disku a uživatel k nim přistupuje pomocí různých síťových služeb a aplikací. Jaký souborový systém použít už záleží na potřebných faktorech, jako můžou být maximální velikost souboru, maximální délka názvu souboru, kódování znaků v adresářích a snadná rozšiřitelnost. Uvedené vlastnosti jsou limitovány, a tak je vhodné provést analýzu, kterou je zjištěno, jaký souborový systém bude pro dané použití nejvhodnější. Mezi nejznámější síťové souborové systémy patří *NFS*, *CODA*, *GFS*, *Samba* a *Kerberos*.

3.5 Server

Jednou z nejdůležitějších částí aplikace pro synchronizace dat přes internet je serverová část a zajištění jejího chodu. Aplikace by měla být složena minimálně ze dvou částí, kde první části je prostor pro uložení dat a druhou je aplikace běžící na serveru, která řeší synchronizaci. Aby bylo možné službu určitým způsobem provozovat a uživatelé měli možnost registrace, tak je nutné mít na serverové části i nějaké uživatelské prostředí, kde bude umožněno vytvoření účtu, možnost stáhnutí klientské aplikace a kde mohou být o aplikaci přečtené užitečné informace. S provozem na serveru je nutné vyřešit otázku na konkrétní požadavky a předejít možným problémům, popřípadě vědět, jaký má problém řešení a jak ho vyřešit. Nejčastějšími možnými problémy můžou být:

- **Přetečení bufferů.**
- **Přetížení a dostupnost serveru.**
- **Dostupnost aplikace.**

3.6 Nastavení firewallu a NATu

Při distribuci dat v sítí je možné použít různé bezpečnostní prvky, kterými je například *firewall* nebo *NAT*, který navíc umožňuje připojit k internetu více počítačů, pod jednou *IP adresou*. S použitím těchto prvků jsou ovšem naskytnuty problémy, které jsou dále popsány. Problémy mohou vzniknout vždy, pokud je nějaká služba špatně nastavená.

Firewall je schopen blokovat, cokoli je potřeba, aby se někdo nedostal, kam nemá, popřípadě je schopen zabránit různým útokům z venku. Lze jím zablokovat například *IP adresu* nebo kterékoliv porty. Při distribuci dat můžou být problémem právě zablokované (zavřené) porty, a tak data nebudou moci kolovat jakýmkoliv směrem. Je-li tedy firewall špatně nastaven, tak může být běžná komunikace vyhodnocena jako útok a bude zablokována. Je tedy dbát na správné nastavení *firewallu.*

NAT slouží pro přeložení adresy z vnitřního adresového rozsahu do veřejného a naopak, což je prováděno na směrovači. Z tohoto principu má použití *NATu* výhodu v tom, že případný útočník nezná strukturu sítě, na kterou chce zaútočit nebo nemůže dojít ke spojení přímo s konkrétním počítačem. Problém s použitím *NATu* například je, že směrovač musí sledovat všechna spojení a vědět o všech zabraných portech a pokud o tom všem vědět nebude, tak spojení může zamrznout, což je běžný problém *FTP serverů*, proto je nutné věnovat pozornost nastavení *NATu*.

4 Implementace aplikace *PersonalSync*

Součástí této kapitoly bude popis mnou vytvořené aplikace, její funkčnost a popis zvolených rozhodnutí při mém vývoji. Název aplikace vychází z názvu této bakalářské práce a pojmenoval jsem jí *PersonalSync*, proto bude dále pro vytvářenou aplikaci používán pouze tento název.

4.1 Použité technologie

Pro implementaci jsem použil všechny technologie, jež jsou dostupné pod licencí *open-source* a zcela zdarma. Není tedy potřeba pořizovat nějakou placenou licenci některého frameworku či databáze. Vývojová prostředí pro implementaci jsou rovněž volně dostupné a plně vyhovující pro vývoj.

4.1.1 Java

Pro aplikaci jsem použil objektově orientovaný programovací jazyk *Java* [15]. Tento jazyk jsem použil z důvodu velkého rozšíření u uživatelů po celém světě a navíc mi nabízí řadu možností a spoustu funkcí pro různá řešení. Pro aplikaci *PersonalSync* je důležitá práce síťová komunikace a práce s databází, což *Java* umožňuje. *PersonalSync* je složena ze tří naprogramovaných částí a všechny jsou napsány v *Javě*, čímž mi odpadly možné problémy komplikovaných komunikací mezi různými technologiemi. Pro veškerý vývoj, který souvisí s *Javou,* jsem použil vývojové prostředí *Eclipse* [16].

4.1.2 Swing

Klientská aplikace je implementována v Javě pomocí knihovny *Swing* [17], jenž obsahuje základní komponenty potřebné pro tvorbu uživatelského rozhraní. U klientské části je nutností komunikovat se serverem, což je potřebné pro aplikaci *PersonalSync*. Podmínkou pro funkčnost uživatelského programu je mít nainstalovanou *Javu* (*JVM*) v počítači uživatele této aplikace.

4.1.3 Vaadin

Pro webovou část, kde je uživatel seznámen s používáním *PersonalSync* jsem použil framework *Vaadin* [18], který vyvinula finská společnost *IT MILL* od roku 2002 pod stejným názvem projektu. Projekt byl v roce 2007 znovu přejmenován na *IT MILL Toolkit* a až v roce 2009 získal projekt svůj současný název *Vaadin. Vaadin* je moderní webový framework, pro tvorbu uživatelského rozhraní na webu, který je poměrně nový. Komponenty ve *Vaadinu* jsou uživatelsky velmi přívětivé a kód je velmi podobný *Swingu* s tím rozdílem, že výsledná aplikace neběží jako desktopová aplikace, ale běží přímo v internetovém prohlížeči u uživatele. Píšeme-li aplikaci ve *Vaadinu*, tak píšeme serverovou část.

API Vaadinu je založeno na *GWT* a obsahuje téměř všechny komponenty, které jsou používány ve webových aplikacích a postačují na bohatou webovou aplikaci. Existují ovšem doplňující komponenty jako jsou například speciální grafy nebo kalendáře, které nejsou součástí standardního *API* balíčku. To už je považováno za velký nadstandard, protože standardní *API* je již tak velmi obsáhlé. Při psaní webové aplikace ve *Vaadinu* je psán kód (uživatelské rozhraní, logika) pouze v *Javě,* který je pomocí GWT přeložen do *JavaScriptu* a pomocí něho je aplikace zobrazovaná v internetovém prohlížeči. Po zkompilování zdrojového kódu je vytvořen *WAR* balíček, který stačí pouze nahrát na webový

aplikační server, v případě *PersonalSync* to je *Tomcat*. Výslednou aplikaci je možno nasadit takřka na každém aplikačním serveru, který umí spolupracovat s konfiguračním souborem *web.xml* pomocí něhož je výsledná aplikace nahraná na aplikační server a je na něm spuštěna.

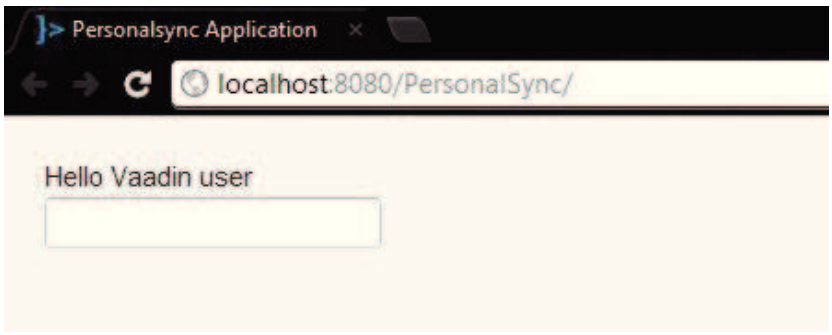
Vyvíjet aplikaci ve *Vaadinu* je poměrně jednoduché a nejpoužívanější prostředí (*Eclipse, NetBeans*) nabízí integraci doplňku pro tvorbu *Vaadin projectu*, které vytvoří přímo spustitelný projekt. Stačí mít nainstalovaný nějaký webový server a po vytvoření *Vaadin projectu* je automaticky vytvořena hlavní třída, kterou je spuštěna celá aplikace, jak je vidět na ukázce.

```
import com.vaadin.Application;
import com.vaadin.ui.*;

public class PersonalSyncApplication extends Application {
	@Override
	public void init() {
		Window mainWindow = new Window("Personalsync Application");
		Label label = new Label("Hello Vaadin user");
		TextField textfield = new TextField();
		mainWindow.addComponent(label);
		mainWindow.addComponent(textfield);
		setMainWindow(mainWindow);
	}
}
```

Výpis 1: Spouštěcí třída Vaadin aplikace

Hlavní aplikace musí dědit z třídy *Application*, kde jsou řešeny základní funkce pro chod aplikace. Ze zdrojového kódu je patrné, že po spuštění aplikace je zavolána metoda *init()*, v níž je vytvořeno hlavní okno aplikace a do něho jsou postupně přidány dvě komponenty *Label* a *TextField*. Uvedený příklad po spuštění vypadá, tak jak je zobrazeno na následujícím obrázku.



Obrázek 1: Základní Vaadin aplikace v prohlížeči

Vzhled komponent *Vaadinu* nemusí být líbivý pro každého, a tak existují dnes už témata, které lze jednoduše naimportovat do projektu a použít hotová řešení. Ovšem je i možnost upravit defaultní *CSS styly* od *Vaadinu,* ale nedoporučoval bych to, protože z vlastní zkušenosti vím, že *Vaadin* má komponenty odladěné na míru a při velkém nebo následném rozšiřování aplikace můžou nastat nemalé problémy v rozvržení stránky, proto jsem v aplikaci vytvořil vlastní styl a použil přímo ten. To je provedeno vytvořením třídy v *CSS souboru,* které nastavíme potřebné vlastností, a pak je tato třída zavolána na jakoukoli komponentu, kterou *Vaadin* podporuje. Přiřazení stylu k dané komponentě je provedeno zavoláním metody *setStyleName.* Například *label.setStyleName("myCssClass")*.

4.1.4 Tomcat

Pro serverovou část, kde může uživatel vytvořit účet, stáhnout aplikaci a zjistit základní informace je použit framework *Vaadin*. Finální projekt ve *Vaadinu* je obsažen v balíčku *WAR* a pro ten jsem použil aplikační server *Tomcat* [19], který je pod licencí *open-source*.

4.1.5 MySQL

Soubory jsou v aplikaci *PersonalSync* ukládány do relační databáze a zvolil jsem databázi *MySQL* [20], protože podporuje uložení datového typu *BLOB* pro synchronizované soubory a je rovněž pod licencí *open-source* do jisté míry. Jelikož není potřeba nadstandardních funkcí, tak *MySQL* je plně dostačující pro mou potřebu. Pro vývoj a administraci databáze jsem používal aplikaci *MySQL Workbench.*

4.1.6 XML

Technologii *XML* [21] jsem použil pro uložení struktury souborů a složek na disku. *XML* je značkovací jazyk, pomocí něhož je možno přehledně uchovávat data v souboru s příponou *XML*. Data jsou v *XML* dokumentu uchovávaná dvěma způsoby. Buď jsou součástí elementů, nebo jsou zapsána jako atributy. Výhodou *XML* je přehledné zobrazení dat uživateli, jejich zobrazení a zápis.

4.2 Rozdělení aplikace

Aplikace *PersonalSync* je složena ze 4 částí a to databáze, klientské aplikace a dvou samostatných komponent běžících na serveru, které jsou dále popsány a na obrázku 2 znázorněny vztahy jednotlivých komponent.

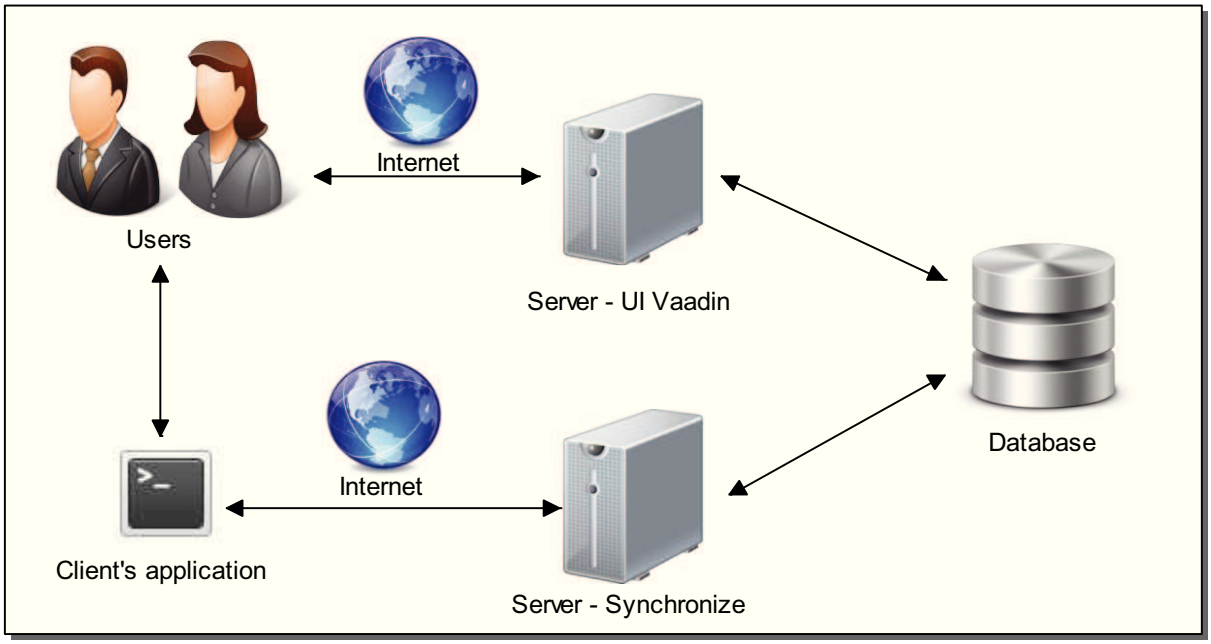
Pro používání aplikace je zapotřebí, aby uživatel nejprve vytvořil účet. K tomu jsem vytvořil webovou aplikaci ve *Vaadinu*, jež je možno umístit na kterýkoliv server podporující *WAR package*. Tato aplikace je samostatná část a je napojena na databázi, kde je vytvořen příslušný účet, pokud jsou splněny podmínky pro vytvoření.

Po vytvoření účtu uživatel může stáhnout klientskou aplikaci, což je další komponenta celé aplikace *PersonalSync.* Klientská aplikace je program napsaný v *Javě,* díky kterému jsou pomocí jednoduchého uživatelského rozhraní provedeny veškeré důležité nastavení pro synchronizaci dat. Program komunikuje s druhou serverovou částí, díky které je řešena synchronizace.

Druhou serverovou částí a zároveň třetí je aplikace běžící na serveru, která komunikuje s klientskou aplikací. I tato serverová část je napojena na databázi, kterou využívá podle potřeby. Je to prostředník mezi klientskou aplikací a databází. Od klienta přijímá požadavky a ty zpracovává. Požadavky od klienta souvisí vždy s daty v databázi, a jelikož k datům nelze z bezpečnostních důvodů přistupovat přímo z klientské aplikace, tak bylo nutné vytvořit mezivrstvu. Požadavky na tuto část souvisí s ověřováním přihlašovacích údajů, posílání jednotlivých souborů a případná změna uživatelského hesla.

Čtvrtou a poslední části je databáze, na kterou jsou napojeny obě serverové částí. V prvním případě s ní pracuje komponenta napsaná ve *Vaadinu*, kdy vytváří nový účet a v druhém případě s ní pracuje *Java* server, přes který proudí soubory od klienta do databáze a zpět.

Na obrázku 2 je grafický znázorněno, s kým jednotlivé komponenty spolupracují. Je patrné, že uživatel přijde do styku s dvěma komponentami. První je webová aplikace, pomocí níž může vytvořit účet a druhou je klientská aplikace, která komunikuje s druhou serverovou části. Aby uživatel mohl využívat kteroukoli serverovou část, tak musí být připojen k internetu, což je z na obrázku 2 patrné. Díky rozdělení na dvě serverové části je možné použít tyto serverové aplikace odděleně. Výhodou tohoto rozdělení je, že v případě, kdy jeden ze dvou serverů odpadne, tak druhý stále poběží. Pokud ovšem spadne databáze, tak obě serverové částí poběží, ale budou v tuto chvíli nevyužité, protože veškera komunikace potřebuje spolupracovat s databází, která nebude dostupná. Je samozřejmě možné nasadit oba servery a databázi nasadit na jeden server nebo to různě zkombinovat, ale to záleží na konkrétním požadavku. S tímto rozdělením je nabízená otázka, proč při synchronizaci komunikuje klient přes server a nekomunikuje přímo s databází. Je to proto, protože kdyby komunikoval přímo s databází, tak nám kdokoli může nabourat databázi a poškodit ji, protože by bylo možné zjistit přihlašovací údaje z klienta. Tímto řešením je tomuto zamezeno a komunikace s databází probíhá přes server.

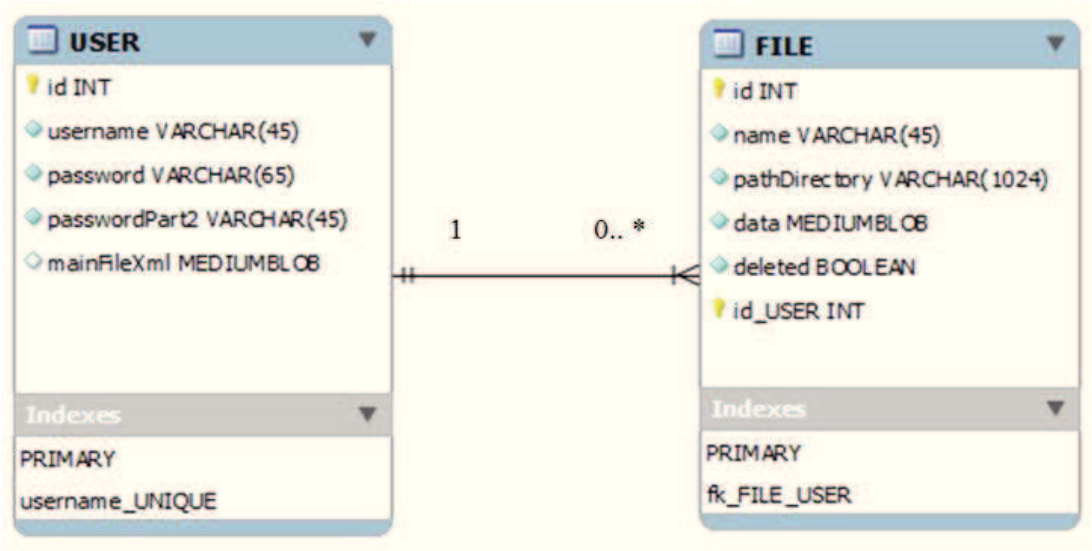


Obrázek 2: Rozdělení aplikace

4.2.1 Databáze

Systémem řízení báze jsem zvolil *MySQL*. V aplikaci *PersonalSync* je nutnosti mít údaje o uživatelích a jeho souborech. K tomu stačí mít v databázi pouze dvě entity. První entitou je pochopitelně uživatel a jednotlivé soubory jsou ukládány do druhé tabulky souborů. Může vypadat, že takto zvolena databáze v případě jediné tabulky pro soubory může být špatnou volbou, jelikož uživatel může mít mnoho souborů a v případě mnoho uživatelů by vyhledáváních jednotlivých souborů mohlo být pomalé, ale dnešní technologie jsou dost rychle a jsou schopny pracovat s velkým objemem dat. Navíc lze v databázi oindexovat důležité atributy a zvýšit tak výkon vyhledávání a manipulaci požadovaných položek, proto jsem použil takové řešení. Vztah mezi těmito tabulkami popisuje následující obrázek 3, na němž je zobrazen ER diagram, kdy jeden uživatel může mít žáden libovolný počet souborů.

4.2.1.1 Entity a atributy



Obrázek 3: ER Diagram

Tabulka *USER* obsahuje pouze 5 atributů. Jediným indexovaným atributem je primární klíč *id* uživatele. Význam jednotlivých atributů je dále popsán.

- **Id** – jedinečný identifikátor uživatele číselného typu uložen v databázi jako *Integer*.
- **Username** – je jedinečné uživatelské jméno s maximálním počtem znaků 45 zapsáno *VARCHAR(45)*, pomocí něhož je uživatel registrován k používání aplikace. Uživatelským jménem musí být validní emailová adresa, aby bylo zabráněno nahodilému vytváření uživatelských účtů a zbytečnému zabírání místa v databázi, protože jeden uživatel může vlastnit jeden účet a s případnými problémy lze uživatele ověřit pomocí emailové adresy, zda je to opravdu on. Uživatele lze vytvořit pomocí aplikace přímo a nekontroluje se, zda emailová adresa fyzicky existuje, ale do budoucna je možno aplikaci o tuto funkcionalitu rozšířit.
- **Password** – je heslo uživatele s maximálním počtem znaků 65 a opět je hodnota datového typu *VARCHAR*. Heslo má zvýšenou bezpečnost, aby byl ztížen možný slovníkový útok. Heslo je vytvořeno na základě funkce: *password = MD5(MD5(passOfUser) + passwordPart2)*, kde *password* je hodnota uložena v databázi. Heslo uživatele *passOfUser* je nejprve zahašované metodou *MD5* a k této hodnotě je přidána hodnota *passwordPart2* tzv. *sůl*, v mém případě to je čas v milisekundách, kdy byl účet vytvořen, což je popsáno u následujícího atributu. Tyto dvě hodnoty jsou opět zahašovany, což tvoří výsledné heslo.
- **PasswordPart2** – slouží pro uchování času v milisekundách, kdy byl účet vytvořen. Je využíváno pro získání výsledné hodnoty hesla. Představuje hodnotu *passwordPart2* ve vzorci *password = MD5(MD5(passOfUser) + passwordPart2)*. Tato hodnota je uchovávaná v databázi pomocí datového typu *VARCHAR*.
- **MainFileXml** – je místo, pro soubor *Xml*, který jsem použil pro strukturu souborů a složek uvnitř vybrané složky pro synchronizaci. Tento soubor vytváří klientská aplikace, která ho potřebuje a pracuje s ním v závislosti na požadavcích uživatele. Pro tento soubor byl použit datový ty*t MEDIUMBLOB*, který může poskytnout velikost až necelých 17 MB, přesněji

16777215 bajtů. Předcházející velikosti datových typů, které by bylo možné v *MySQL* použít mohou mít velikost až 255 bajtů *(TINYBLOB*) nebo 65535 bajtů *(BLOB*), což by mohlo být poměrně málo. V případě potřeby je možnost použít i datový typ *LONGBLOB,* který umožňuje uložit soubor až 4 GB, ale takové struktury souborů a složek nejsou zcela běžné a 99% místa by bylo nevyužito v tomto atributu, proto jsem zvolil datový typ *MEDIUMBLOB*.

V tabulce *FILE*, je 6 atributů, z nichž jsou 3 oindexované. Prvním indexovaným je atribut *id_USER*, což je cizí klíč z tabulky *USER*. Dále je indexováno *id,* které je klíčem tabulky *FILE* a posledním indexovaným je atributem je *name*, protože soubory budu podle tohoto atributu velmi často vyhledávat.

- **Id** – jedinečný identifikátor pro každou entitu. Zvoleným datovým typem je *BIGINT*.
- **Name** – pro jméno souboru je použít *VARCHAR* s maximální přípustnou délkou 260 znaků, což je i maximální délka v *OS Windows* a například pro osoby používající tuto aplikaci s tímto systémem by byla přípustná větší délka zbytečná, navíc možnost použít až 260 znaků je myslím postačující. Atribut *name* představuje název souboru spolu s jeho příponou.
- **PathDirectory** – představuje cestu k danému souboru v podobě datového typu *VARCHAR*. Vyzkoušel jsem na několika počítačích, jak hluboké jsou nejdelší zanoření od kořene uživatelových souborů a složek a maximální délka ve všech případech nepřesáhla 300 znaků. Uživatelé jsou ovšem různí a můžou mít různé struktury na disku, proto byla maximální hodnota zvolena 1024 znaků. Pokavaď bude mít ovšem zanoření souborů nebo složek delší než je zvolené maximum, tak soubory nebudou synchronizovány. Bylo možné zvolit cestu k souboru společně s názvem, ale takový návrh mi umožňuje lépe pracovat a mít o souborech lepší přehled.
- **Data** – pro uložení souboru jako binárních dat slouží tento atribut a lze do něho uložit soubor o maximální velikosti 16777215 bajtů, což je necelých 17 MB.
- **Deleted** – tento atribut je datového typu *BOOLEAN* a signalizuje příznak, zda je soubor smazán nebo ne.
- **Id_USER** – představuje v tabulce *FILE* cizí klíč tabulky *USER* a parametry jsou proto totožné, jako má tento atribut v tabulce *USER.*

4.2.1.2 Datové slovníky

USER						
Atribut	**Datový typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
id	INT	2^32 - 1	ANO	NE	ANO	Id uživatele
username	VARCHAR	45	NE	NE	NE	Uživatelské jméno
password	VARCHAR	65	NE	NE	NE	Heslo
passwordPart2	VARCHAR	45	NE	NE	NE	Časové razítko
mainFileXml	MEDIUMBLOB	2^24 - 1	NE	ANO	NE	Soubor struktury

Tabulka 3: Datový slovník pro tabulku USER

FILE						
Atribut	**Datový typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
id	LONG	2^64 - 1	ANO	NE	ANO	Id souboru
name	VARCHAR	260	NE	NE	ANO	Název souboru
pathDirectory	VARCHAR	1024	NE	NE	NE	Cesta k souboru
data	MEDIUMBLOB	2^24 - 1	NE	NE	NE	Data souboru
deleted	BOOLEAN	2	NE	ANO	NE	Příznak smazání
id_USER	INT	2^32 - 1	NE	NE	ANO	Id uživatele

Tabulka 4: Datový slovník pro tabulku FILE

4.2.1.3 Rozhraní JDBC

JDBC [22] je známo svou rychlostí při manipulaci s daty, ale na úkor této rychlosti je náročnější vyvíjet aplikaci z časového hlediska, kdy je stráveno více času na vývoji projektu při složité doméně, než když je použit nějaký framework. V aplikaci *PersonalSync* je očekáván velký přenos dat, proto jsem použil v aplikaci *JDBC,* čímž je zpomalení možného frameworku částečně eliminováno a vzhledem k tomu, že je aplikace složena pouze ze dvou entit na straně databáze, tak v databázi nejsou žádné složité modely. Použitím *JDBC* mi nebude tedy dělat zas takový problém případné rozšiřování.

Pro používání *JDBC* s *MySQL* databázi je potřeba použít na obou serverových částech konektor, pro připojení. V *PersonalSync* jsem použil konektor *mysql-connector-java-5.0.8-bin.jar*, který byl stáhnut ze stránek používané databáze [20] a je součástí přílohy.

4.2.1.4 Nastavení

Pro vytvoření databáze je k této práci dodán SQL skript s názvem *CreateDatabase.sql,* pomocí kterého lze vytvořit schéma v databázi. Schéma je nazváno podle názvu aplikace *PersonalSync*. Pro připojení k databázi je ve zdrojových souborech nastavené přihlašovací jméno na *root* a heslo na *pass.* Při vytvoření databáze je tedy nutné brát ohled na toto nastavení, popřípadě heslo ve zdrojových souborech (*MySQLConnectivity.java*) změnit a následně soubory zkompilovat. Při časté změně těchto údajů je možné k aplikaci vytvořit konfigurační soubory a načítat hodnoty z nich.

Defaultní nastavení databáze je pro aplikaci *PersonalSync* nevyhovující především kvůli přenosu velkých souborů a to až necelých 17 MB. Při vývoji bylo potřeba pro správnou funkčnost aplikace změnit v databázi následující parametry:

- **Delayed_insert_limit –** změněna hodnota 100 na 1200.
- **Query_cache_limit –** zvětšena maximální hodnota dotazu z 1 MB na 2 MB.
- **Max_allowed_packet –** hodnota nastavena z 1 MB na 17 MB (17825792 B).

4.2.1.5 Struktura XML

Pro uložení struktury složek a souborů jsem zvolil uchovávat informace v *XML* souboru, protože je to přehledné a snadno lze zachytit stromovou strukturu umístění jednotlivých souborů a složek. Struktura je zobrazena následujícím výpisem kódu:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SynchronizedFile version="0">
      <file changed="Tue Apr 24 06:33:15 CEST 2012" name="Docs"
            type="folder">
            <file changed="Tue Apr 24 06:41:00 CEST 2012" name="AAA.tex"
                  type="file" />
            <file changed="Tue Apr 03 08:20:53 CEST 2012" name="BBB.pdf"
                  type="file" />
      </file>
      <file changed="Tue Apr 03 08:10:41 CEST 2012" name="Photos"
      type="folder" />
      <file changed="Tue Apr 05 04:10:53 CEST 2012" name="CCC.docx"
            type="file" />
      <file changed="Tue Apr 05 04:10:25 CEST 2012" name="DDD.docx"
            type="file" />
</SynchronizedFile>
```

Výpis 2: Příklad struktury XML

Kořenový uzel má název *SynchronizedFile* a obsahuje pouze jeden atribut. Tím atributem je *version*, což uchovává verzi souborů, kolikrát byla jedním uživatelem editována a synchronizována jeho vybraná složka určena pro synchronizaci. Pomocí tohoto atributu je zjišťováno, zda je současná verze aktuální a na základě porovnání pak probíhají jednotlivé synchronizace.

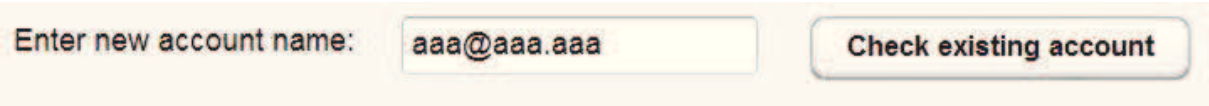
Ze struktury souboru XML je vidět, že všechny soubory a složky mají své vlastnosti v atributech elementu <*file*>. Adresář a soubor je pak rozlišován na základě atributu *type*, který může nabývat pouze hodnot *file* nebo *folder*. Rozdíl mezi těmito dvěma hodnotami je patrný již podle názvu. Pokud má atribut hodnotu *folder*, což představuje složku, tak uvnitř složky mohou být další soubory a složky a element *folder* pak bude uzavřen až za všemi obsahujícími soubory a složkami. Pokud je hodnota atributu type *file,* tak tímto je vyznačen soubor a element je ihned uzavírán. Složka může být také prázdná a v tomto případě je pak element s hodnotou atributu type *folder* uzavřen ihned. Ve výpisu 2 je prázdnou složkou například složka *Photos.* Umístění souborů ve složce je ve výpisu 2 znázorněno soubory *AAA.tex* a *BBB.pdf*, které jsou ve složce *Docs.*

Dalším atributem je *name,* což je název souboru a posledním atributem je *changed*, který může sloužit pro porovnávání, kdy je pomocí tohoto atributu možné porovnávat jednotlivé soubory. Je možné zde použít i jiný atribut, jako může být haš hodnota souboru apod. V aktuální verzi aplikace je tento parametr nepoužíván, a tak je připraven pro následující rozšíření. Zanoření jednotlivých souborů a složek je získáno průchodem celého souboru, protože navržena struktura XML představuje umístění souborů a složek na disku.

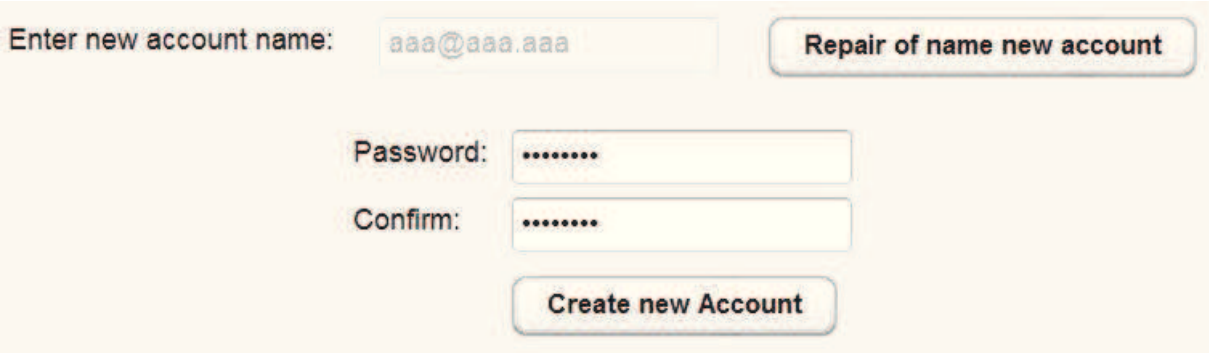
4.2.2 Komponenta PersonalSync Vaadin

Jak už bylo výše zmíněno, pomocí této komponenty mohou zájemci o aplikaci vytvořit účet, jelikož tato komponenta je určena pro veřejné stránky aplikace *PersonalSync*, kde je napsána i nápověda k aplikaci. Uživatel na webu zadá možné uživatelské jméno v podobě emailové adresy, a pokud uživatelské jméno neexistuje, tak může zadat 2x heslo a tím je případnému zájemci vytvořen účet v databázi a pak už stačí jen stáhnout klienta z těchto stránek, kde je umístěn odkaz pro stažení.

Na obrázku 4 je možné vidět, jak vypadají komponenty pro vytvoření účtu. Po kliknutí na tlačítko *Check existing account* je aplikace dotazována v databázi, zda účet existuje, či nikoliv. Pokud účet neexistuje, tak uživateli vyskočí dvě nové políčka (obrázek 5), kde je potřeba 2x zadat stejné heslo a pokud jsou hesla stejné, tak aplikace vytvoří v databázi účet. Poté už stačí jen stáhnout klienta, z odkazu, který je uveden na stránce. Po ověření existence účtu, je políčko pro nový účet zablokované a nelze ho změnit. V tuto chvíli ještě není heslo vytvořeno, a tak může nastat situace, že kdokoli jiný zadá v tuto chvíli stejné uživatelské jméno, tak mu to bude povoleno. Účet je pak vytvořen pouze tomu, kdo klikl dřív na tlačítko vytvoření. Ten co klikne později, tak bude upozorněn hláškou, že nastala chyba, a ať je opraveno uživatelské jméno. Bude-li ovšem opravdový vlastník emailové adresy, tak je potřeba kontaktovat podporu aplikace, že není možné vytvořit účet a nejspíš někdo zneužil emailovou adresu. Do budoucna je tedy možné aplikaci rozšířit o autorizaci uživatelských jmen pomocí ověřovacích kódu zaslaných na emailovou adresu.



Obrázek 4: Vytvoření účtu



Obrázek 5: Vytvoření hesla

Tato komponenta je rozdělena do tří hlavních balíků a na stejné úrovni jako tyto balíky je třída *PersonalSyncVaadin.class,* pomocí které je spouštěna aplikace na aplikačním serveru. Balíky jsou pojmenovány *common*, *controller* a *model*.

4.2.2.1 Balíček *common*

Generalenum balíček – je balíček obsahující logiku pro enumy. Součástí balíčku je 1 rozhraní (*GenericEnum.class*), jež implementuje každý *enum*. Pak je součástí 5 enumů z kterých jsou volány textové hodnoty v aplikaci. Byl kladen důraz na to, ať je aplikace co nejvíc flexibilní a nejsou v kódu psány přímo textové hodnoty, proto jsou veškeré textové hodnoty v *enumech* a je-li potřeba změnit například popisek tlačítka *Check existing account* (obrázek 4), tak je hodnota změněna pouze v *enumu ButtonsEnum* a není zasahováno do kódu aplikace, kde volání hodnoty z *enumu* vypadá, jak je uvedeno v následujícím výpisu:

```
btnExistsAccount = new Button(ButtonsEnum.BTN_EXISTS_ACCOUNT.getValue());
```

Výpis 3: Volání *enumu* v kódu

Výhoda použití *enumů* je patrná. Při jakékoliv změně víme, kde bude hodnota uložena a kde jí je možno editovat, než složitě hledat v kódu. Kód je takto mnohem přehlednější a enumy lze používat kdekoli a kolikrát bude potřeba.

EmailValidator.class – třída, pomocí které je prováděna validace emailu.

MD5Hash.class – třída, která hašuje heslo pro uložení do databáze.

MySQLConnectivity.class – třída, zajišťující připojení do databáze. V této třídě je volán konektor *JDBC* pro použitou *MySQL* databázi, který je uveden v kapitole 4.2.1.3.

4.2.2.2 Balíček *controller*

CheckExistingAccount.class – obsahuje logiku po stisknutí tlačítka, zda existuje účet v databázi.

CreateNewAccount.class – třída, pomocí které je vytvářen účet v databázi po kliknutí na tlačítko *Create New Account*.

4.2.2.3 Balíček *model*

User.class – třída, která reprezentuje uživatele.

UserDAO.class – třída, která je používána pro pracování s objektem *User* na straně databáze.

4.2.2.4 Třída *PersonalSyncVaadin.class*

Na stejné úrovni výše uvedených balíčků je *PersonalSyncVaadin.class*, v níž je vytvořeno uživatelské rozhraní a aplikace je zde i spouštěna.

4.2.3 Komponenta PersonalSync Client

K synchronizaci dále potřebuje uživatel komponentu *PersonalSync Client*. Ve finální fázi je tato komponenta složena ze dvou částí. První je spustitelný *JAR* balíček (*PersonalSyncClient.jar*) a druhou je soubor *cfg.properties*, kde jsou uloženy důležité informace, které program potřebuje pro svou funkčnost. Podmínkou pro správnou funkčnost je mít nutnost mít soubory ve stejném adresáři. Program je napsán v uživatelském rozhraní *Swing* a pro funkčnost je zapotřebí mít nainstalovanou *Javu* v počítači. Výhodou celého řešení je odpadnutí jakékoliv instalace.

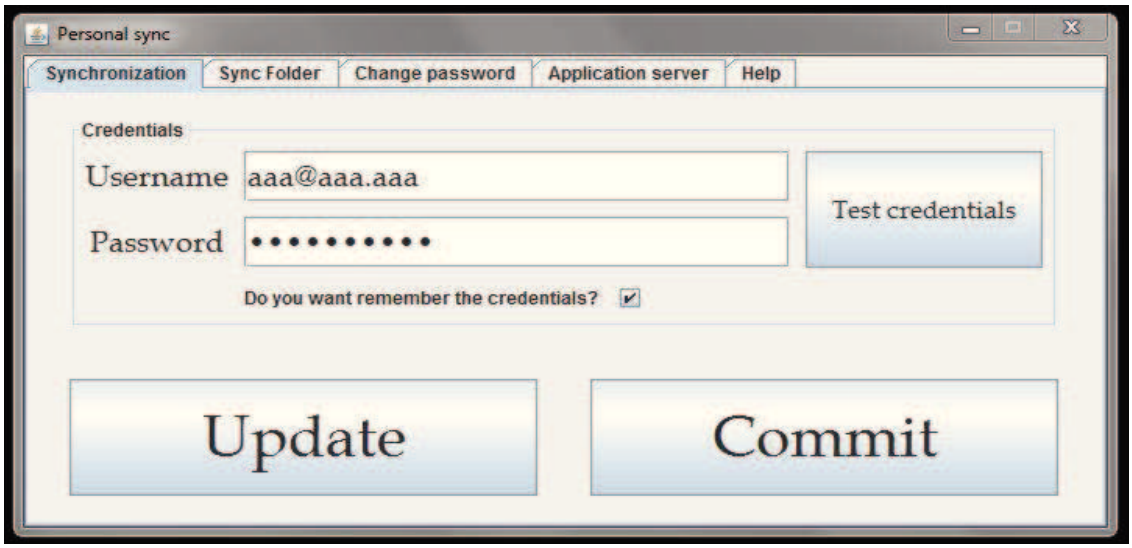
Soubor *cfg.properties* je součástí stahovaného klienta a pokud oba soubory budou ve stejném adresáři, tak aplikace bude pracovat korektně. V opačném případě bude uživatel informován, že tento soubor chybí a program nebude spuštěn. Soubor *properties.cfg* byl použit pro uložení nastavení aplikace, čímž je aplikace flexibilnější a není potřeba při každé změně kompilovat zdrojový kód. Hodnoty v tomto souboru lze měnit pomocí klientské aplikace a ukládané atributy jsou dále popsány v jednotlivých bodech:

- **Password** – uchovává heslo v zahašované hodnotě.
- **PathOfSyncFile** – uchovává cestu synchronizované složky. Důležitý parametr, protože v každém počítači může být cesta jiná, stačí byť jen jiný název uživatele.
- **RememberCredentials** – může nabývat pouze hodnot *true* nebo *false*, který signalizuje, zda mají být uložené přihlašovací údaje. Pokud jsou údaje uloženy, tak využito je pouze přihlašovací jméno. Po spuštění aplikace jsou hodnoty políček *username* a *password* naplněny, ale heslo je v podobě haše, čili heslo je nutné zadat vždy pro lepší bezpečnost. Soubor *cfg.*properties je možné snadno otevřít, a tak by kdokoli mohl heslo získat, jestliže by měl přístup k počítači, proto toto opatření.
- **CumputerName** – obsahuje název počítače, ve kterém je aktuální klient. Zkopíroval-li by někdo aplikaci do jiného počítače, tak aplikace nemusí fungovat korektně, proto je při každém spuštění tato hodnota kontrolována a při případném problému je uživatel informován, ať změní synchronizovanou složku, protože spouští klienta v jiném počítači.
- **Username** – je uživatelské jméno uživatele. Uloženo je pouze tehdy, přeje-li si uživatel údaj uložit.
- **ApplicationServerURL** – obsahuje adresu k serveru, jenž provádí synchronizaci mezi klientem a databází, a který bude umístěn někde na internetu. Adresa musí obsahovat i port, na kterém server běží. Tato hodnota v souboru byla zvolena, protože umístění serverové částí může být často měněno a takto může hodnotu uživatel sám, aniž by stáhnul novou verzi konfiguračního souboru.

Aplikace je složena z 5 komponent *JTabbedPane*, které slouží jako záložky v horní části aplikaci. Navigace je tedy jednoduchá a podle názvu je umožněno snadné přesunutí na požadovanou záložku v aplikaci. Každá záložka má svou funkcí a jsou dále popsány na následujících obrázcích.

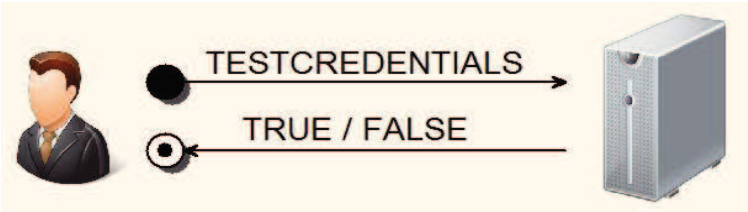
Hlavní okno aplikace je zobrazeno na obrázku 6. Při spuštění klientské aplikace je zkontrolován název počítače (*ComputerName*), a pokud byla aplikace dříve použita na jiném počítači, tak uživatel je o tom informován a přesunut na záložku, kde je možnost změny složky pro synchronizaci. Dále je zkontrolován parametr *RemembersCredentials*, zda mají být načteny přihlašovací údaje a podle nastavení jsou načtena nebo ne. Pokud údaje mají být načteny, tak uživatelské jméno je načteno v pořádku, ale heslo je načteno zašifrováno. Heslo je tedy potřeba vždy zadat z důvodu lepšího zabezpečení. Další

možnostmi na této záložce je otestování přihlašovacích údajů, obnovení synchronizované složky ze serveru tlačítkem *Update* nebo promítnutí aktuálního stavu synchronizované na server tlačítkem *Commit*. Poslední možnou akcí na této záložce je možnost uložení uživatelského jména, ať není potřeba toto jméno zadávat znova při dalším spuštění.



Obrázek 6: Hlavní záložka *Synchronization*

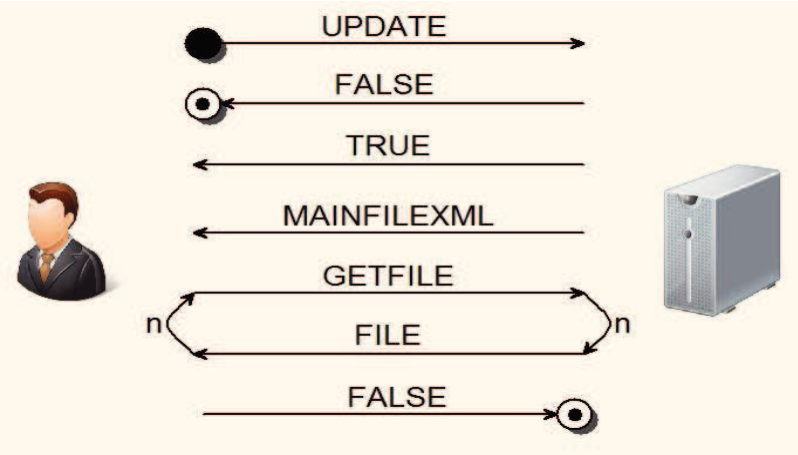
Akce pod tlačítky už potřebuji ke komunikaci server. Pro tuto komunikaci byl vytvořen vlastní protokol, který je závislý na přijatých a odeslaných zprávách. Akce pod tlačítkem *Test credentials* testuje uživatelské jméno a heslo. Uživateli je následně oznámen výsledek v podobě vyskakující zprávy. Z obrázku 7 je možné vyčíst průběh komunikace, kdy jsou na server poslány potřebné informace pro otestování přihlašovacích údajů a jako první odeslána zpráva je textová hodnota *TESTCREDENTIALS*, podle které je přizpůsobeno chování na straně serveru. Komunikace končí ve chvíli, kdy klient obdrží zprávu *TRUE* nebo *FALSE*.



Obrázek 7: Akce *Test credentials*

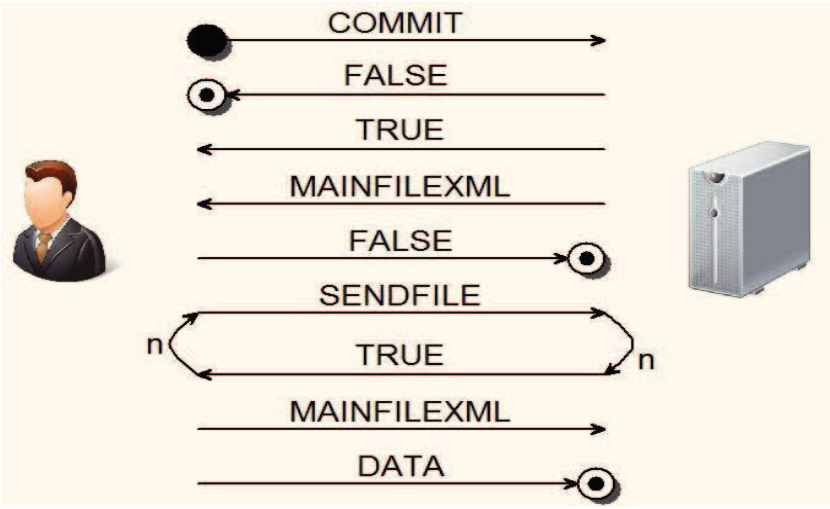
Další možnou akcí je klik na tlačítko *Update.* Po stisknutí je složka aktualizována a jsou do počítače nahrány soubory ze serveru a složka je tak synchronizovaná. Složky není potřeba nahrávat, protože nejsou nositelem dat. Složky jsou vytvořeny na základě *XML* souboru a až do nich jsou poté umísťovány soubory. Akce začíná posláním přihlašovacích údajů na server společně s určením typu akce, jež je v tomto případě *UPDATE* (obrázek 8). Pokud jsou uživatelské údaje nekorektní nebo soubor *mainFile.xml* neexistuje, tak je uživateli vraceno *FALSE* a komunikace končí. V opačném případě je vráceno *TRUE*, za čímž ihned následuje poslání hlavního souboru *mainFile.xml*, kde je uložena struktura souboru a složek. Následně je zkontrolována verze souboru, a pokud je atribut version v *XML* souboru stejný, tak klient posílá serveru zprávu *FALSE* a komunikace je tímto ukončena.

V opačném případě je u klienta prvně smazán obsah synchronizované složky a jsou vytvořeny složky podle přijatého *XML* souboru. Dále je po serveru od klienta požadováno zprávou *GETFILE o* zasláni souborů na základě *XML*. Soubor je pak z databáze vytažen na základě obsahu zprávy *GETFILE*, která obsahuje název souboru a umístění v adresáři vůči synchronizované složky. Tento proces je opakován n krát když $n \in (0, \infty)$. Komunikace je ukončena posláním zprávy *FALSE* od klienta.



Obrázek 8: Akce *Update*

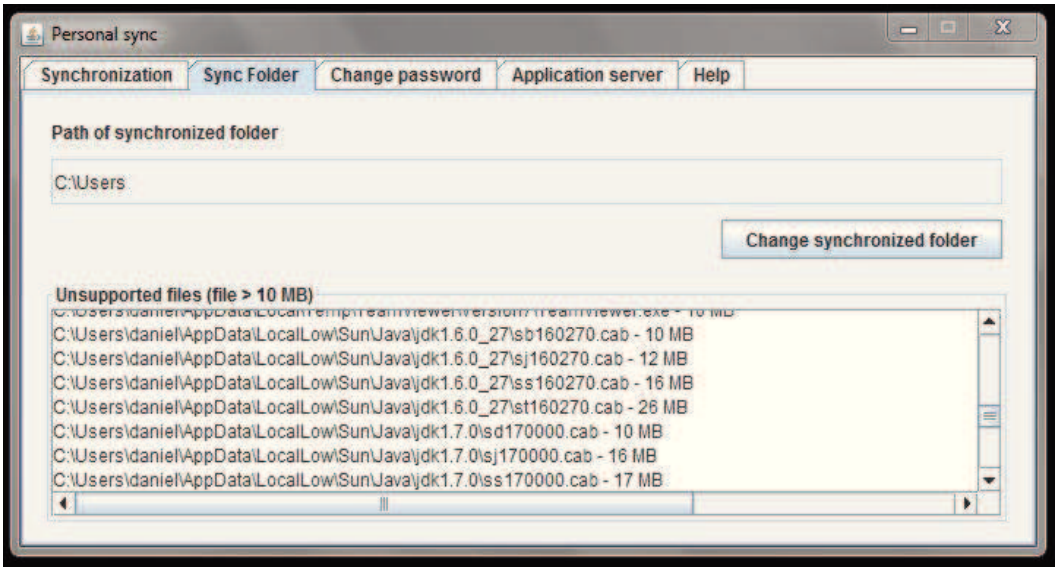
Pod tlačítkem *Commit* je ukryta funkcionalita pro přenesení aktuálního stavu synchronizované složky na server. Celou akci popisuje obrázek 9. Klient posílá zprávu *Commit* společně s přihlašovacími údaji. Pokud údaje nesedí, proces je ukončen posláním zprávy *FALSE* klientovi. V opačném případě server posílá zprávu *TRUE* a za touto zprávou následuje soubor obsahující strukturu složek a souborů. Po kontrole verzí u klienta může klient poslat zprávu *FALSE*, pokud jsou verze stejné, jinak je poslána serveru zpráva *SENDFILE* a server přijímá n souborů, když $n \in (0, \infty)$. Komunikace končí zasláním zprávy *MAINFILEXML* a následným přijetím souboru uchovávající strukturu souborů a složek od uživatele. Na straně serveru je po prvním přijetí zprávy *SENDFILE* nastaven u všech uživatelových souborů atribut *deleted* na *true*, čímž jsou soubory nahrané v této akci jako aktuální.



Obrázek 9: Akce *Commit*

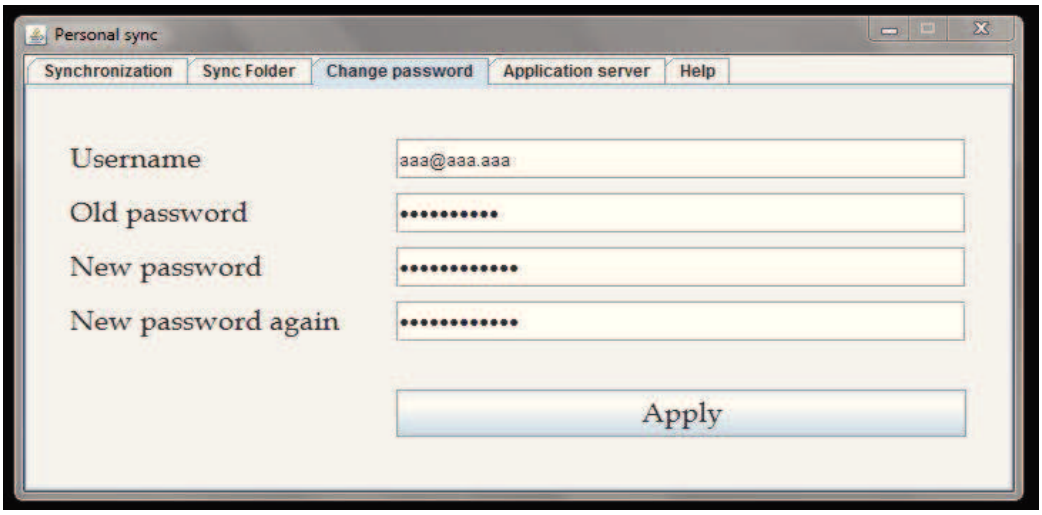
Poslední možností na hlavním tabu (obrázek 6) je zaškrtnutí políčka pro zapamatování uživatelských údajů. Údaje jsou načteny ze souboru *cfg.properties* nebo jsou do něho uloženy ihned po zaškrtnutí respektive odškrtnutí.

Další záložkou je *Sync Folder* (obrázek 10), jež je prostor, kde uživatel může změnit synchronizovanou složku. V dolní části je prostor pro zobrazení souborů v této složce, které jsou větší, než 10 MB, a které nejsou podporovány pro synchronizaci. Naplnění tohoto prostoru probíhá vždy spuštěním nového vlákna při spuštění klienta nebo při změně synchronizované složky. Nové vlákno je použito, protože v hlavním vlákně běží uživatelské rozhraní a aplikace by působila při načítání těchto souborů neovladatelně, protože by nebylo možné udělat jakoukoli akci po dobu načítání.



Obrázek 10: Záložka Sync Folder

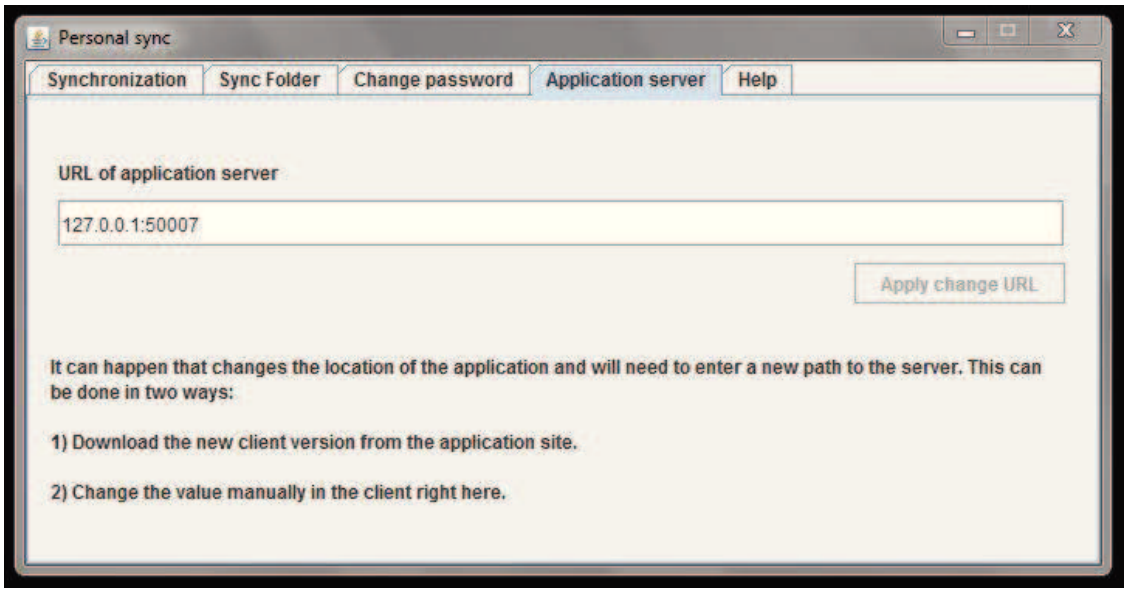
Třetí záložkou v pořadí je záložka *Change password* (obrázek 11) pro změnu stávajícího hesla. Údaje jsou odeslány, pouze pokud je emailová adresa validní a pokud je nové heslo v obou případech stejné, ať není zatěžována síť. Hesla pro možnou změnu jsou posílaná již zašifrována z důvodu bezpečnosti.



Obrázek 11: Záložka *Change password*

Komunikace po stisku tlačítka *Apply* je téměř stejná, jako je tomu v případě otestování přihlašovacích údajů (obrázek 7) s tím rozdílem, že jako typ akce je zaslána hodnota *CHANGECREDENTIALS* a přidán jeden parametr navíc, čímž je zahašovaná hodnota nového hesla.

Předposlední záložkou klientské aplikace je záložka *Application server* (obrázek 12), kde je možné změnit adresu k serveru. Tato záložka byla vytvořena z důvodu možné změny umístění serveru, aby klient nemusel při každé změně stahovat nová nastavení (*cfg.properties*) pro klientskou aplikaci. Potvrdit lze pouze validní *IP* adresu nebo validní doménové jméno (*http://localhost:50007*) a je potřeba zadat i port, na kterém aplikace běží. V defaultním nastavení je adresa nastavená na *127.0.0.1*, což představuje adresu *localhostu*. Port pro aplikaci byl zvolen *50007*. Tento port byl vybrán na základě volných portů [23] pro soukromé využití. Využít lze porty v rozmezí (49152 - 65535). Pokud by tento port využívala jiná služba, tak v této záložce to lze snadno změnit, ovšem je potřeba změnit i serverovou část.



Obrázek 12: Záložka Application server

Poslední záložkou je záložka *Help*, kde jsou pouze uvedeny možné postupy řešení různých problémů. Při jednotlivých akcích po stisknutí každého tlačítka jsou všechny tlačítka zablokována, dokud neskončí akce, čímž jsou následně povolena. Pokud jsou tedy tlačítka zablokována, tak to znamená, že aplikace pracuje a je nutné vyčkat na jejich povolení či informativní hlášku. Pokud jsou tlačítka odblokována, znamená to, že akce je ukončena.

Komponenta *PersonalSyncClient.jar* je na nejvyšší úrovni rozdělena do 4 balíčku a společně s nimi je na stejně spouštěcí třída aplikace třída *Main*.

4.2.3.1 Balíček *Common*

GeneralEnum balíček, EmailValidator.class, MD5Hash.class – plní stejnou funkcí jako v první popsané komponentě s tím rozdílem, že v tomto balíčku *generalEnum* je enumů 9.

CFGProperties.class – třída sloužící pro načítání hodnot ze souboru *cfg.properties.* Třída dědí ze třídy *Properties.class* a byla mnou upravena do podoby návrhového vzoru *Singleton.*

MessageWindow.class – Při spuštění klientské aplikace je zavolána tato třída, která obsahuje pouze jednu statickou metodu, pro snadnější volání informačních zpráv kdekoliv v aplikaci.

4.2.3.2 Balíček *Action*

Součástí tohoto balíčku je dalších 6 balíčku spojených s akcemi po stisku jednotlivých tlačítek.

TabMain balíček – obsahuje třídy *TestCredentialsClick.class, UpdateClick.class, CommitClick.class* a *CheckRememberClick.class*. Tyto třídy řeší logiku stisků jednotlivých tlačítek na záložce *Synchronization.* Kontrolují správně vyplněné údaje a je-li vše v pořádku, tak spouští nové vlákno pomocí třídy *ServerConnection.class*, která bude dále popsána.

TabSyncFolder balíček – obsahuje třídu *FillingAreaThread.class*, která po změně složky nebo spuštění aplikace prochází v novém vlákně synchronizovanou složku a obrazuje uživateli nepodporované soubory pro synchronizaci. Druhou třídou v tomto balíčku je *SelectSyncFolder.class*, pomocí které je možné změnit synchronizovanou složku.

TabChangePassword balíček – obsahuje pouze jednu třídu *ChangePasswordClick.class*, která vytváří nové vlákno pomocí třídy *ServerConnection.class*, jsou-li údaje pro změnu hesla vyplněny korektně.

TabApplicationServerUrl balíček – obsahuje třídu *CheckTheSameURL.class*, která povoluje stisknutí tlačítka, je-li nová adresa serveru odlišná od původní. Druhou třídou v tomto balíčku je *ApplyChangeURL.class*, která řeší logiku uložení nové adresy do konfiguračního souboru *cfg.properties.*

Network balíček – obsahuje důležitou třídu *ServerConnection.class*, která je novým vláknem aplikace a řeší částečně spojení připojení k serveru. V této třídě je přetěžován konstruktor a následně podle typu akce vytvoří jednu ze tříd *TestingCredentials.class, Updating.class, Comitting.class* nebo *ChangingCredentials.class*, které jsou taktéž v tomto balíčku. Tyto třídy pak komunikují se serverem a řeší příslušnou logiku komunikace a posílání zpráv.

XmlMapping balíček – obsahuje třídy *FileForSync.class, MapFileAndFolder.class, MappingFileAndolder.class* a *XmlWriter.class*, které řeší veškerou práci s *XML* souborem uchovávající strukturu složek a souborů.

4.2.3.3 Balíček *Model*

Obsahuje pouze třídy *FileSync.class* a *User.class* představující model entit v databázi.

4.2.3.4 Balíček *UI_Swing*

Zde je vytvářeno uživatelské rozhraní obsahující na stejné úrovni třídu *Gui.class* a balíček *tab*.

Tab balíček – obsahuje třídy *TabHelp.class*, *TabMain.class*, *TabSyncFolder.class*, *TabChangePassword.class* a *TabApplicationServerURL.class* , které vytváří uživatelské rozhraní pro jednotlivé záložky. Poslední třídou v tomto balíčku je abstraktní třída *AbstractCommonProperties.class*, z které dědí některé záložky za účelem jednotných fontů písem.

Gui.class – třída která řeší základní nastavení uživatelského rozhraní. Dále jsou zde přidávány všechny záložky v klientské aplikaci.

4.2.3.5 Třída *Main.class*

Třída, v níž je spouštěna aplikace obsahující jedinou metodu *main*.

4.2.3.6 Chybové stavy

Při používání klientské aplikace může docházet k různým problémům, které jsou v aplikaci ošetřeny, a uživateli je problém oznamován vyskakující zprávou v novém okně, kde může vyčíst i případné řešení nastalého problému. Některé problémy už byly zmíněny, ovšem ve skutečnosti může dojít k mnoha dalším chybovým stavům, na které je uživatel upozorněn a které jsou dále popsány.

- **Chybějící soubor *cfg.properties*** – pokud uživatel spouští klientskou aplikaci a nemá ve stejném adresáři, jako je spustitelný *JAR* klient soubor *cfg.properties*, tak program není spuštěn.
- **Nevybrána složka pro synchronizaci a nevyplněná adresa serveru** – při prvním spuštění klienta na počítači, je klient vyzván, ať vybere složku pro synchronizaci a nastaví adresu aplikačního serveru. Server může být už nastaven po stažení, a tak při vyplněné položce serveru není třeba nic vyplňovat. Pokud uživatel nenastaví tyto dvě hodnoty, tak nedojde k žádné akci.
- **Chyba na straně databáze** – při jakékoliv chybě v databázi nebo v situaci, kdy databáze nepoběží, tak bude uživatel informován o její nefunkčnosti. Pokud databáze nefunguje již při prvním pokusu o připojení na straně serveru, tak následně při této chybě bude uživatel po každé akci informován zprávou, že přihlašovací údaje nejsou v pořádku, protože nemohly být ověřeny.
- **Chyba připojení k serveru** – tato chyba nastane, pokud nebude spuštěn server nebo adresa k serveru bude chybná. Uživatel bude vyzván o případnou opravu adresy.
- **Chybná adresa serveru** – adresu serveru je možné zadat dvěma způsoby. Buď v podobě *IP* adresy, nebo v podobě klasického doménového jména. V každém případě musí být zadán i port, na kterém aplikace běží, jinak adresa serveru nebude uložena a uživatel bude vyzván o nápravu.
- **Validní vstupní údaje** – po stisku na kterékoliv tlačítko je potřeba pracovat s uživatelským jménem. Na první záložce *Synchronization* je navíc potřeba zadat heslo. Pokud nejsou vždy vyplněny oba údaje, tak je uživatel požádán o jejich vyplnění. Pokud údaje jsou vyplněny, tak je navíc kontrolována emailová adresa, zda je validní. Pokud validní není, tak je uživatel vyzván o nápravu. Na záložce *Change password* je potřeba vyplnit 3 různé údaje. Uživatelské jméno, stávající heslo a 2x nové heslo. Zde jsou kontrolovány opět všechny položky, zda jsou vyplněné, jinak akci pro změnu hesla nebude možné provést. Opět je zde kontrolována validní emailová adresa a navíc jsou zde kontrolovány obě položky pro nové heslo. Nové heslo je potřeba potvrdit dvojitý zadáním, a pokud údaje nejsou shodné, tak akce není provedena a uživatel je požádán o nápravu.
- **Správné přihlašovací údaje** – při každé akci jsou potřeba přihlašovací údaje. Pokud ověřené údaje nejsou správné, tak je uživateli oznámeno, že údaje nejsou v pořádku.
- **Ostatní chyby** – ostatní chybové jsou spojeny s jednotlivými akcemi a uživateli je vždy v krátké zprávě řečeno, kde nastala případná chyba a co dělat.

4.2.4 Komponenta PersonalSync Server

Poslední komponentou je *PersonalSyncServer.jar*, což je serverová část umístěna mezi klientskou aplikací a databázi. Aplikací je možné spustit kdekoli bude nainstalovaná *JVM* a zároveň musí mít počítač veřejně dostupnou adresu, aby bylo možné připojení k této aplikaci. Aplikace je možné spustit kliknutím na spustitelný soubor, ale lepší variantou je spustit aplikaci přes příkazovou řádku, kde je možné sledovat aktuální požadavky různých klientů. V operačním systému *Windows 7* je příkaz pro spuštění následující: *java –jar PersonalSyncServer.jar.* Server běží na portu *50007,* jak bylo výše uvedeno. Tato komponenta je o něco menší než klientská aplikace a obsahuje následující balíčky, jež jsou dále popsány.

4.2.4.1 Balíček *Common*

GeneralEnum package, MySQLConnectivity.class – plní stejnou funkcí, jako v předešlých komponentách. V tomto balíčku jsou enumy pouze 3.

4.2.4.2 Balíček *Action*

Obsahuje 5 tříd, kde je řešena logika po jednotlivých akcích uživatele. Jsou to třídy *TestCredentials.class, TestingCredentials.class, UpdatingServer.class, CommittingServer.class* a *ChangingPassword.class.* Při každé akci je nutné ověření a k tomu slouží třída *TestCredentials.class*, ostatní třídy v tomto balíku tuto třídu využívají.

4.2.4.3 Balíček *Model*

Obsahuje 4 třídy, z nichž dvě *FileSync.class* a *User.class* představují jednotlivé entity v databázi a ke každé této třídě je třída s koncovkou *DAO.* Tedy *FileSyncDAO.class* a *UserDAO.class*, které komunikují s objekty přímo v databázi. Nastává otázka, proč není třída pojmenována názvem *File*, jako je tomu v databázi. V *Javě* jsem nemohl použít název *File* pro třídu, protože *Java* již obsahuje třídu s takovým názvem, proto jsem třídu pojmenoval *FileSync.class*.

4.2.4.4 Třída *Server.class*

Hlavní třída, která je spuštěna po startu serveru, jež naslouchá na zvoleném portu. Po přijetí požadavku vyváří tato třída nové vlákno, které představuje třída *MiniServerThread.class*.

4.2.4.5 Třída *MiniServerThread.class*

Tato třída je volána po přijetí požadavku uživatele na server. Podle příslušné akce rozhoduje, která třída v kapitole 4.2.4.3 je vytvořena, v níž je řešena logika požadavků. Po ukončení akce je zde spojení i ukončeno.

5 Závěr

Cílem bakalářské práce bylo analyzovat problémy při synchronizaci dat a naimplementovat vlastní aplikaci, která bude data synchronizovat. Vývoji předcházela analýza současných aplikací na trhu, které nemusí každému vyhovovat, a tak jsem přišel s vlastním řešením.

Vývoj aplikace *PersonalSync* splnila svůj účel, protože byla vyvinuta aplikace, která synchronizuje veškeré soubory, jelikož jsou v aplikaci synchronizovány soubory na základě binárních dat. Aplikace je v současné verzi určena především pro synchronizaci menších souborů a využití najde tam, kde jsou synchronizována data, jež jsou z velké části měněná. *PersonalSync* nabízí zajímavý pohled na synchronizaci, kdy je struktura adresářů a složek uchovávaná v souboru *XML*. Navíc aplikaci není nutné potřeba vůbec instalovat a umožňuje pouze vlastní režii synchronizace dat, kdy jsou data synchronizována až na požadavek uživatele.

Do budoucna je aplikací možné rozšířit z mnoha pohledů. Například při vytváření účtu je možné ověřovat emailovou adresu zasláním ověřovacího kódu na email, který uživatel zadá při vytváření a účet by byl v databázi vytvořen až po tomto ověření. Z pohledu synchronizace je možné porovnávat různé parametry a na základě nich synchronizovat pouze tyto soubory a snížit zatížení sítě nebo by bylo možné k aplikaci vytvořit i webové rozhraní, pomocí něhož by uživatel mohl spravovat své soubory přímo na webu.

Daniel Žažo

6 Reference

[1] Dropbox [online]. [cit. 2012-04-20]. Dostupné z: https://www.dropbox.com/

[2] SugarSync [online]. [cit. 2012-04-20]. Dostupné z: https://www.sugarsync.com/

[3] Synplicity [online]. [cit. 2012-04-20]. Dostupné z: http://www.syncplicity.com/

[4] Windows Live Mesh [online]. [cit. 2012-04-20]. Dostupné z: http://windows.microsoft.com/cs-CZ/windows-live/essentials-other-programs

[5] Data synchronization. [online]. [cit. 2012-04-20]. Dostupné z: http://en.wikipedia.org/wiki/Data_synchronization

[6] TRIDGELL, Andrew. Efficient Algorithms for Sorting and Synchronization [online]. [cit. 2012-04-20]. Dostupné z: http://www.samba.org/~tridge/phd_thesis.pdf. PhD thesis. The Australian National University.

[7] Kryptografie a počítačová bezpečnost (KPB). Eliška Ochodková [online]. [cit. 2012-04-20]. Dostupné z: http://www.cs.vsb.cz/ochodkova/

[8] Simplified DES. CryptoGraphy - University of Rhode Island [online]. [cit. 2012-04-20]. Dostupné z: http://www.cs.uri.edu/cryptography/dessimplified.htm

[9] C.A. VAN TILBORG, Henk a Sushil JAJODIA. Encyclopedia of Cryptography and Security [online]. [cit. 2012-04-20]. ISBN 978-1-4419-5905-8. Dostupné z: http://books.google.cz/books?id=UuNKmgv70lMC&printsec=frontcover&hl=cs&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[10] BIRYUKOV, Alex Biryukov a Dmitry KHOVRATOVICH. Related-key Cryptanalysis of the Full AES-192 and AES-256 [online]. [cit. 2012-04-20]. Dostupné z: http://eprint.iacr.org/2009/317.pdf. University of Luxembourg.

[11] RIVEST, Ron, Adi SHAMIR a Leonard ADLEMAN. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: [online]. [cit. 2012-04-20]. Dostupné z: http://people.csail.mit.edu/rivest/Rsapaper.pdf

[12] ŠAFRÁNKOVÁ, Jana. Zálohování dat v domácím prostředí [online]. [cit. 2012-04-20]. Dostupné z: http://dspace.upce.cz/bitstream/10195/37599/1/SafrankovaJ_Zalohovani%20dat_MN_2010.pdf. Bakalářská práce. Univerzita Pardubice Fakulta ekonomicko-správní.

[13] IBM Informix Database Design and Implementation Guide. Systém nápovědy IBM [online]. [cit. 2012-04-20]. Dostupné z: http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.ddi.doc/ddi164.htm

[14] A Low-bandwidth Network File System. In: MUTHITACHAROEN, Athicha, Benjie CHEN a David MAZIERES. [online]. [cit. 2012-04-20]. Dostupné z: http://pdos.csail.mit.edu/papers/lbfs:sosp01/lbfs.pdf

[15] Java [online]. [cit. 2012-04-20]. Dostupné z: http://www.java.com/en/

[16] Eclipse [online]. [cit. 2012-04-20]. Dostupné z: http://www.eclipse.org/

[17] Swing [online]. [cit. 2012-04-20]. Dostupné z: http://docs.oracle.com/javase/7/docs/

[18] Vaadin [online]. [cit. 2012-04-20]. Dostupné z: https://vaadin.com/home

[19] Tomcat [online]. [cit. 2012-04-20]. Dostupné z: http://tomcat.apache.org/

[20] MySQL [online]. [cit. 2012-04-20]. Dostupné z: http://www.mysql.com/

[21] XML [online]. [cit. 2012-04-20]. Dostupné z: http://www.w3schools.com/xml/

[22] Studijní materiály k předmětu DAIS: Databázové a informační systémy [online]. [cit. 2012-04-20]. Dostupné z: http://dbedu.cs.vsb.cz/SubPages/Courses/Course.aspx?course=dais&year=2011-2012

[23] Service Name and Transport Protocol Port Number Registry. [online]. [cit. 2012-04-27]. Dostupné z: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml

Přílohy

A Obsah DVD

Obsah jednotlivých adresářů obsažených v přiloženém DVD:

bin - obsahuje spustitelné verze vyvíjeného softwaru

- *PersonalSyncVaadin.war* – balíček určen pro serverovou část, kde se vytváří účet
- *PersonalSyncClient* – složka v níž je spustitelný klient *PersonalSyncClient.jar* a konfigurační soubor v defaultním nastavení *cfg.properties*
- *PersonalSyncServer*.jar – spustitelný *JAR* soubor určen pro serverovou, kde je řešena synchronizace

src – obsahuje zdrojové kódy vyvíjeného softwaru

- *PersonalSyncVaadin* – složka obsahující zdrojový kód veřejné serverové částí
- *PersonalSyncClient* – složka obsahující zdrojový kód klientské části
- *PersonalSyncServer* – složka obsahující serverové části pro synchronizaci
- *CreateDatabase.sql* – skript pro vytvoření databáze

support – obsahuje doprovodný software

- *MySQL Connector/J* – oficiální *JDBC* ovladač pro *MySQL*
- *MySQL WorkBench* – instalační soubor pro administraci databáze *MySQL*
- *Tomcat* – instalační soubor služby *Java* webového serveru

text – obsahuje text bakalářské práce

- *PersonalSync.pdf* – práce ve formátu *PDF*
- *UserManual.pdf* – uživatelský manuál ke klientské aplikaci ve formátu *PDF*
- *PersonalSync.docx* – práce ve formátu *DOCX*
- *UserManuel.docx* – uživatelský manuál ke klientské aplikaci ve formátu *DOCX*